AD-A268 705

NAVAL POSTGRADUATE SCHOOL
Monterey, California

DTIC
ELECTE
SEP 0 1 1993
S A D

THESIS

COMPUTER-AIDED PROTOTYPING SYSTEM (CAPS) WITHIN THE SOFTWARE ACQUISITION PROCESS: A CASE STUDY

by

Mary Kay Ellis

June, 1993

Thesis Advisor: Luqi

Approved for public release; distribution is unlimited.

93-20430

93 8 31 139

REPORT DOCUMENTATION PAGE

1a Report Security Classification: Unclassified		1b Restrictive Markings	
2a Security Classification Authority		3 Distribution/Availability of Report	
2b Declassification/Downgrading Schedule		Approved for public release; distribution is unlimited.	
4 Performing Organization Report Number(s)		5 Monitoring Organization Report Number(s)	
6a Name of Performing Organization Naval Postgraduate School	6b Office Symbol *(if applicable)* 39	7a Name of Monitoring Organization Naval Postgraduate School	
6c Address *(city, state, and ZIP code)* Monterey CA 93943-5000		7b Address *(city, state, and ZIP code)* Monterey CA 93943-5000	
8a Name of Funding/Sponsoring Organization	6b Office Symbol *(if applicable)*	9 Procurement Instrument Identification Number	
Address *(city, state, and ZIP code)*		10 Source of Funding Numbers	
		Program Element No / Project No / Task No / Work Unit Accession No	

11 Title *(include security classification)* COMPUTER-AIDED PROTOTYPING SYSTEM (CAPS) WITHIN THE SOFTWARE ACQUISITION PROCESS: A CASE STUDY

12 Personal Author(s) Ellis, Mary, K

13a Type of Report	13b Time Covered From To	14 Date of Report *(year, month, day)*	15 Page Count
Master's Thesis		June 1993	114

16 Supplementary Notation The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

17 Cosati Codes			18 Subject Terms *(continue on reverse if necessary and identify by block number)*
Field	Group	Subgroup	Computer-Aided Prototyping; CAPS; Software Acquisition.

19 Abstract *(continue on reverse if necessary and identify by block number)*
This thesis provides a case study which examines the benefits derived from the practice of computer-aided prototyping within the software acquisition process. An experimental prototyping system currently in research is the Computer Aided Prototyping System (CAPS), managed under the Computer Science department of the Naval Post Graduate School, Monterey, California. This thesis determines the qualitative value which may be realized by applying the Computer-Aided Prototyping System (CAPS) to the initial stages of the acquisition process for a software system on the scale of a prototype model, then projecting the results to a real-time distributed computer system. As a prelude to this analysis, information is presented concerning how the acquisition process is currently managed within DoD and what role prototyping plays within that process. An introduction to the CAPS is then given, along with a description of its capabilities obtained through personal examination of the system. Following this walkthrough of the CAPS, software acquisition is discussed further, including an analysis of its major obstacles and where a CAPS could best be used within the acquisition cycle. This thesis concludes with a cost analysis and results from a comparison of performing a requirements analysis and feasibility study with and without a CAPS.

20 Distribution/Availability of Abstract	21 Abstract Security Classification	
__ unclassified/unlimited __ same as report __ DTIC users	Unclassified	
22a Name of Responsible Individual Luqi	22b Telephone *(include Area Code)* 408-656-2735	22c Office Symbol CS/Lq

UNCLASSIFIED

Approved for public release; distribution is unlimited.

Computer-Aided Prototyping System (CAPS)
Within the Software Acquisition Process:
A Case Study

by

Mary K. Ellis
Captain, United States Air Force
B.S., Pennsylvania State University, 1984

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN (Systems Technology)

from the

NAVAL POSTGRADUATE SCHOOL
June 1993

Author: Mary Kay Ellis
Mary K. Ellis

Approved by:
Luqi, Thesis Advisor

David A. Gaitros, Co-Advisor

Paul H. Moose, Chairman
Command, Control, and Communications Academic Group

ABSTRACT

This thesis provides a case study which examines the benefits derived from the practice of computer-aided prototyping within the software acquisition process. An experimental prototyping system currently in research is the Computer Aided Prototyping System (CAPS), managed under the Computer Science department of the Naval Post Graduate School, Monterey, California. This thesis determines the qualitative value which may be realized by applying the Computer-Aided Prototyping System (CAPS) to the initial stages of the acquisition process for a software system on the scale of a prototype model, then projecting the results to a real-time distributed computer system.

As a prelude to this analysis, information is presented concerning how the acquisition process is currently managed within DoD and what role prototyping plays within that process. An introduction to the CAPS is then given, along with a description of its capabilities obtained through personal examination of the system. Following this walkthrough of the CAPS, software acquisition is discussed further, including an analysis of its major obstacles and where a CAPS could best be used within the acquisition cycle.

This thesis concludes with a cost analysis and results from a comparison of performing a requirements analysis and feasibility study with and without a CAPS.

Availability Codes

Dist | Avail and/or Special

A-1

TABLE OF CONTENTS

LIST OF FIGURES

I. INTRODUCTION

A. OVERVIEW

This thesis provides a case study which involves the usage of computer-aided prototyping for accurate requirements definition in support of the software acquisition process. Specifically, this thesis determines the qualitative value which may be realized by applying the Computer-Aided Prototyping System (CAPS) to the process of coding a software system on the scale of a prototype model, then projecting the results to a real-time distributed computer system. The CAPS is managed under the Computer Science department of the Naval Postgraduate School, Monterey, California.

B. RESEARCH QUESTIONS

The primary research question is "What impact would the application of CAPS have upon the requirements process within the acquisition arena for a real-time distributed computer system?" Subsequent questions are:

1. "How is the acquisition process currently managed within DoD for software systems and how does prototyping improve weak points that are encountered when using the traditional acquisition methodology?"
2. "What experimental prototypes have already been produced through the use of the CAPS?"
3. "What are the major obstacles within software acquisition and what portion of the cycle experiences the most difficulty?"

4. "Where would CAPS be used, in what part of the cycle, and by what organization?"

5. "Would the CAPS reduce acquisition costs of a candidate system enough to justify its application as an automated tool for enhancement of DoD software acquisition and maintenance programs?"

C. DISCUSSION

Computer-aided prototyping, which seeks to automate early design phases, could be a useful technique during development or enhancement of software systems, even those which are mission-oriented with time constraints imposed upon them. One area of importance could be that of cost savings. Some of the more costlier mistakes within the software acquisition cycle may evolve from poor requirements definition during the initial phase of the cycle, which then results in an inaccurately coded product and in turn the costly practice of recoding the software to meet the revamped requirements. Another area of importance is the communication which prototyping encourages between the systems analyst and the product user which helps ensure adequate formulation and assessment of system requirements.

An experimental prototyping system currently in research is the Computer Aided Prototyping System (CAPS). It is envisioned that an implemented CAPS will one day support the rapid prototyping of more complex, mission-critical software systems as a tool with visual graphics capability mapped to a program specification language which in turn generates executable Ada code automatically[Ref. 1]. This real-world application of CAPS would aid in improving the traditional software life-cycle through

a two-phase cycle consisting of rapid prototyping and automatic program generation, as well as a support to the system acquisition and integration process. The CAPS has provided the former of the two on a much smaller scale as an experimental model for single-processor target architectures[Ref. 2].

The most highly complex prototype created with the aid of the CAPS to date is that of a command, control, communications, and intelligence (C3I) system. The Generic C3I Workstation prototype has characteristics typical of embedded software, including distributed processing, timing constraints, multiple predefined hardware interfaces, and complex requirements[Ref. 2]. The prototype was developed to run on a Sun 3 and is directly transferable to a ruggedized Genisco computer. An important aside is that the use of the Sun 3 is consistent with possible targeting for future Portable Operating System Interface (POSIX) compliant platforms. The C3I prototype is currently serving as a testbed for ongoing research in computer-aided software design.

This thesis builds upon what has been achieved thus far with CAPS to provide a study of its application to the acquisition requirements process. A comparison of two acquisition methodologies (with and without the use of rapid prototyping) for a scenario involving a software enhancement to one of the prototypes created with the aid of the CAPS (the Generic C3I Workstation) is given with the results projected to a real-world system.

The system chosen for the projection of cost to something of higher complexity is the command and control segment (CCS), a complete set of hardware and software developed by International Business Machines (IBM) Corporation for the Air Force in

support of missions conducted by satellite centers in Sunnyvale, California and in Colorado Springs, Colorado. This system has completed initial development, all audits, has been essentially accepted by the Air Force, and is now in the cycle of software maintenance and modernization upgrades. The system is highly complex with over two million lines of code[Ref. 3]. Its architecture is that of a distributed system of several software modules written mainly in Jovial code[Ref. 3]. There were literally hundreds of software problems existing when the initial system was delivered, and it could not be used operationally for quite some time. Although it is now used in real-time operations, the ongoing maintenance process from validation of requirements to delivery of acceptable code is several months. Because of the environment in which this command and control software is utilized, the mission control room operators cannot wait through a lengthy approval cycle. To conform to user's schedule constraints, temporary software "fixes" for this command and control software are put in place until the final code is delivered. Cost is also a major consideration while attempting to track all software changes for proper specification maintenance which correlates to the operational code. An improved system process is necessary for the acquisition cycle, and CAPS is one tool which may be applied during the requirements definition area of the cycle and possibly in development as well. That is the area of focus for this thesis.[Ref. 3]

II. SOFTWARE ACQUISITION WITH PROTOTYPING FOR ITS ENHANCEMENT

A. OVERVIEW

This chapter presents information and viewpoints from four areas. First, it provides a summarization of the guidance followed by the Department of Defense in managing software development. Second, it provides background information pertaining to the software world within the acquisition process. Third, it discusses prototyping and how it is becoming an integral part of acquisition. Finally, it provides a description of the Computer-Aided Prototyping System as an experimental model of computer-aided prototyping.

B. DEPARTMENT OF DEFENSE GUIDANCE

The Department of Defense utilizes various standards, instructions, and guidelines to aid in the management of the acquisition, operation, and maintenance of software. Documents such as military standards become contractual requirements in acquisition programs. To understand how the DoD conducts these programs, one looks to the requirements spelled out in these documents as discussed below.

1. DoD Directive 8000.1, Defense Information Management Program

The DoD Information Management philosophy is based upon the Goldwater-Nichols Act which promotes "jointness" in the composition of DoD's forces. It is an

outgrowth of two fundamental strategic documents which resulted from an Executive Level Group plan for DoD corporate information management and a Joint Chiefs of Staff (JCS) document specifying the kind of information infrastructure DoD would need in the future.[Ref. 6:p. 2]

The DoD Directive integrates the elements of functional process improvement, information resources management, and information technology and services into a program to manage the full lifecycle of all data and information. Under the first element, senior managers of the Office of the Secretary of Defense and Chairman, JCS now have the responsibility and the authority for functional process streamlining of DoD operations throughout the department. This will provide for common integrated processes throughout the department, integrated across functional boundaries. The second element builds upon an existing operational program, a broader framework of information management. The third element provides information technology and services such as computing, communications, acquisition of information technology components, corporate data and software repositories, and information and systems security through a centrally managed DoD-wide infrastructure. In addition, emphasis is being placed on software reuse and the development and use of methods, models, and tools to accelerate systems development and to facilitate functional and system integration in allowing DoD customers to determine the value of available support.[Ref. 6:p. 2]

2. DoD-Std-2167A, Defense System Software Development

The main government standard referenced for software development (and applicable throughout the life cycle of the software) is DoD-Std-2167A, Defense System Software Development. As well as software in general, this standard also applies to the software element of firmware. It instructs the contractor to perform those software processes which are required by contract, with those which are unnecessary being excluded through contract tailoring. The activities which are executed during this development process and documented by the contractor in a software development plan are[Ref 4:p. 9]:

1. System Requirements Analysis/Design
2. Software Requirements Analysis
3. Preliminary Design
4. Detailed Design
5. Coding and Computer Software Unit Testing
6. Computer Software Component Integration Testing
7. Computer Software Configuration Item Testing
8. System Integration and Testing

For the above activities, the contractor will conduct: software development management, software engineering, software product evaluations, and configuration management, which is discussed in greater detail within the next section.[Ref. 4]

Data requirements to choose from for contractual deliverable include the following: system or system segment design document, software development plan, software requirements specification, interface requirements specification, interface design document, software design document, software product specification, version description document, software test plan, software test description, and software test report. Other deliverables which may be required include: computer system operator's manual, software user's manual, software programmer's manual, firmware support manual, computer resources integrated support document, engineering change proposal, and specification change notice.[Ref. 4:p. 12-13]

This military standard also dictates requirements for software coding standards. It specifies language-independent requirements for coding, which apply to all deliverable source code products developed under the contract.

Presently, a new draft standard is in circulation (Mil-Std-SDD) for replacement of Mil-Std-2167A and also Mil-Std-2168 (Defense Systems Software Quality Program). This new standard will address the procurement, quality, and maintenance of software.[Ref. 8:p. 4]

3. Mil-Std-480B, Configuration Control -- Engineering Changes, Deviations, and Waivers

Another standard for acquisition is Mil-Std-480B, Configuration Control -- Engineering Changes, Deviations, and Waivers. This document establishes the requirements for configuration control as well as the formats and procedures to be utilized in the preparation of configuration control documentation. Included within the

standard are requirements for: maintaining configuration control of configuration items, both hardware and software; preparing and submitting engineering change proposals, requests for deviations and waivers, notices of revision, and specification change notices; and, evaluating, coordinating, and approving or disapproving these documents.[Ref. 9]

This standard is used by contractor and government personnel alike to establish and maintain effective configuration control of the approved configuration identification; for proposing engineering changes to software and hardware; and to control the form, fit, and function of privately developed configuration items. Configuration control changes apply to functional, allocated, or product baseline.[Ref. 9]

4. Mil-Std-490A, Specification Practices

Another military standard within the arena of acquisition is Mil-Std-490A, Specification Practices. This document sets forth practices for the preparation, interpretation, change, and revision of specifications for acquisition programs prepared by or for the departments and agencies of the DoD. This military standard was prepared to establish uniform specification practices in response to the need for a document comparable to that for engineering drawing practices.

Specifications covered by this standard which are prepared as military, federal, contracting agency, or contractor specifications are of the following types: the system or system segment specification, the development specification (which may be a prime item, critical item, non-complex item, facility, or software development specification), the product specification (which may be a prime item function, prime item

fabrication, critical item function, critical item fabrication, non-complex item fabrication, inventory item, or a software product specification), the process specification, and the material specification.[Ref. 10] The type A system/segment specification describes the top-level requirements of the system, including those for software. The type B5 software development specification is a combination of the software requirements specification and the interface requirements specification under DoD-Std-2167A. The type C5 software product specification is a combination of the software design document, interface design document, and source and object code listings of the final software as required under DoD-Std-2167A.[Ref. 11:p. 85]

5. DoD-Std-2168, Defense System Software Quality Program

This military standard describes how to develop, document, and implement a software quality assurance program. Its requirements affect all aspects of the software development effort, including software engineering methodology, production, and testing.[Ref. 11:p. 182]

6. Mil-Std-1521B, Technical Reviews and Audits on Systems, Equipments, and Computer Software

The final military standard to be addressed for acquisition is Mil-Std-1521B, Technical Reviews and Audits on Systems, Equipments, and Computer Software. This document prescribes the requirements for the conduct of technical reviews and audits upon systems, equipments, and computer software. The following technical reviews and audits are selected by the program manager during the appropriate phase of the

program's development: the system requirements review, the system design review, the software specification review, the preliminary design review, the critical design review, the test readiness review, the functional configuration audit, the physical configuration audit, the formal qualification review, and the production readiness review.[Ref. 12]

7. DoD-HDBK-287, A Tailoring Guide for DoD-Std-2167A

Although this handbook is not called out as a contractual requirement for software projects, this guide clarifies requirements in other military standards. It also includes algorithms for applying data item descriptions to a specific project. This helps when attempting to reduce documentation requirements.[Ref. 11:p. 86]

C. SOFTWARE ACQUISITION

The software acquisition cycle is considered to be composed of two segments: software development and software evolution. The discipline common to the total cycle is that of configuration management.

1. Configuration Management

The ability to coordinate software development to minimize difficulties caused by involvement of multiple programmers with the same piece of software is provided through configuration management[Ref. 7:p. 8]. Configuration management is generally divided into four areas: identification, control, status accounting, and audits, both functional and physical. Configuration identification involves the capture of an official software baseline which is comprised of the coded product, procedures, and the documentation that defines that code. Configuration control is the allowance of only

government-authorized changes to the software product and to its documentation, ensuring these changes correspond between product and document, that is, controlling modifications to the software being built by a team of programmers[Ref. 7:p. 8]. Status accounting involves the tracking of authorized changes by maintaining a complete record of data elements to provide a historical record of the product. Physical and functional configuration audits are performed as a verification that the government is indeed receiving all contractually required deliverables which satisfy all physical and performance requirements. The goal of configuration management is to maximize productivity by minimizing mistakes[Ref. 7:p. 8]. Configuration management is normally conducted by both the contractor and the government.

The goals of configuration management in the arena of software evolution include recording the development history of evolving systems, maintaining the integrity of them, and aiding in the management of the systems for controlling their evolution. Module interconnection languages address the integrity of an evolving configuration. Concurrency control is especially important when many designers work simultaneously on different aspects of the same system.[Ref. 13]

The three problems most often encountered which justify the need for configuration management are: the double maintenance problem, the shared data problem, and the simultaneous update problem. The first is encountered when one retains multiple identical copies of software. Updates or changes must be integrated into all copies, which can be easily overlooked. Configuration management supports the avoidance of multiple copies of the same information. The second arises when many

individuals simultaneously access and modify the same data, such as program code. Changes made by one programmer may interfere with the progress of others, such as an invalid modification. Thus, problems are likely to occur when programmers work together on one piece of source code. The third occurs when a software team has difficulty working with one copy of the source code which everyone shares. A solution to this environment is to divide the source code into a number of files or modules. A programmer desiring to make a change to a specific module will modify only a copy of the module initially, will then test the modification, and only when it is a valid change, then implement it into the baseline module. These three are typical coordination problems which occur and are resolved through the assistance of a configuration manager.[Ref. 7:p. 9-15]

2. Software Evolution

Software evolution refers to all activities that change an existing software system[Ref. 5].The common term prior to *evolution* was *maintenance* and refers to those activities within a system's life cycle which follow acceptance by the government for the delivery of the final software product. It involves the product user, acquisition manager, and software engineer in accomplishment of software fixes and upgrades[Ref 13]. An important action in evolution of a large system is ensuring consistency of each new configuration. A problem with changing a component of one's software system is that this may translate into requiring changes in its other components for consistency.[Ref. 13]

3. Software Development

Software development refers to that portion of the acquisition cycle that involves the initial identification of software requirements capabilities from the using organization (operations command) to the design organization (acquisition or materiel command) through which design, test, and delivery of a final product to the government is made[Ref. 4].

The traditional software development model or *waterfall model* (Figure 1) has separate phases covering software requirements analysis, design, implementation, and testing. This approach is adequate for small to medium-scale data processing systems where system requirements, constraints, and functionality are well understood and formulated prior to software design and implementation. However, for the case of large, complex, real-time systems, this approach fails as software requirements cannot be completely or correctly identified unless some part of the system's functionality may be constructed and evaluated.[Ref. 14:p. 1]

A traditional software development model would assume that designers could stabilize and freeze the requirements. However, this may not be completed until users gain experience with the proposed system. Thus, requirements often change after initial implementation. These changes in turn trigger changes within the production version of the system during its maintenance phase. Thus, in prototyping, if a requirement changes, this may trigger changes in the prototype version of the system. However, this is beneficial as a prototype may be modified more easily than a production version.[Ref. 13]

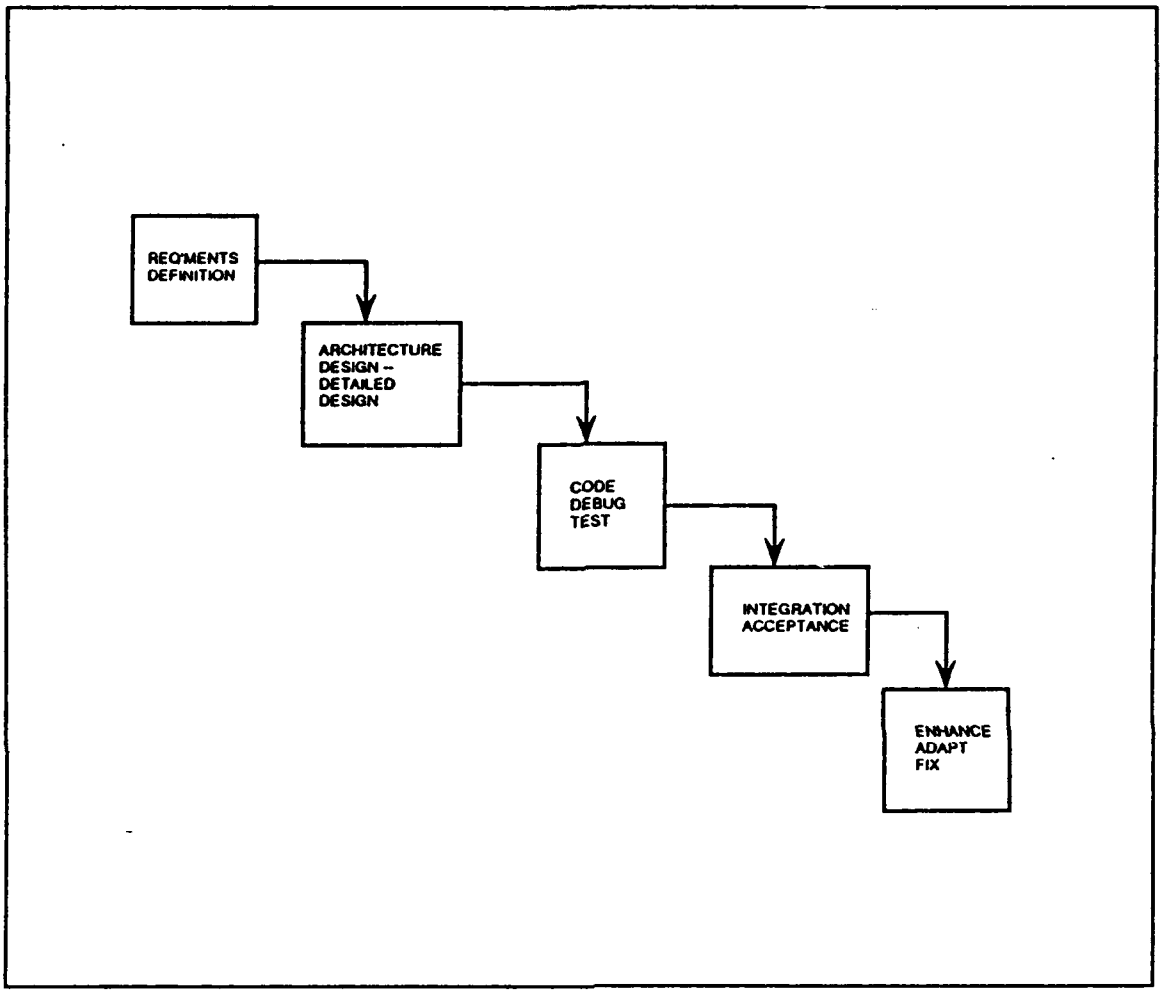

Figure 1 The Waterfall Model

D. THE METHODOLOGY OF PROTOTYPING

For those systems which are not founded on well-defined requirements, software evolution may account for more than half of the total cost of the software. Because corrections and enhancements to a software system may produce a notable amount of cost, automated support for software evolution is a worthy area for attention. This

support is important because evolution based on the bare program code is difficult to achieve. Prototyping provides one approach to achieving the goal of automated support in software evolution.[Ref. 5]

Prototyping has increasingly become an approach adopted to improve the plannability of software projects[Ref. 15:p. 28]. A prototype is an executable model of selected aspects of a proposed system, creating an executable pilot version of the intended system. Rapid prototyping is the building and evaluation of a series of prototypes (Figure 2). It is one method of determining user requirements for software systems. A prototype's usefulness is only as a means to an end. That is, it is created to ensure a user's requirements for his software system are valid; the actual coding itself is seldom used toward the finished product, the real system. Rather, one should consider this "throw-away" code, temporary, used until the requirements definition process is completed. During the process, the user and designer work together to define requirements and specifications for the critical parts of the projected system. During its demonstration, the user evaluates the prototype's actual against expected behavior. The designer uses the validated requirements which result as the basis for designing the production software. Software systems are delivered incrementally and requirements analysis continues throughout the process. Incremental delivery extends the advantages of prototyping to the production environment. Finally, the prototype gives an executable representation of system requirements that may be applied to system testing.

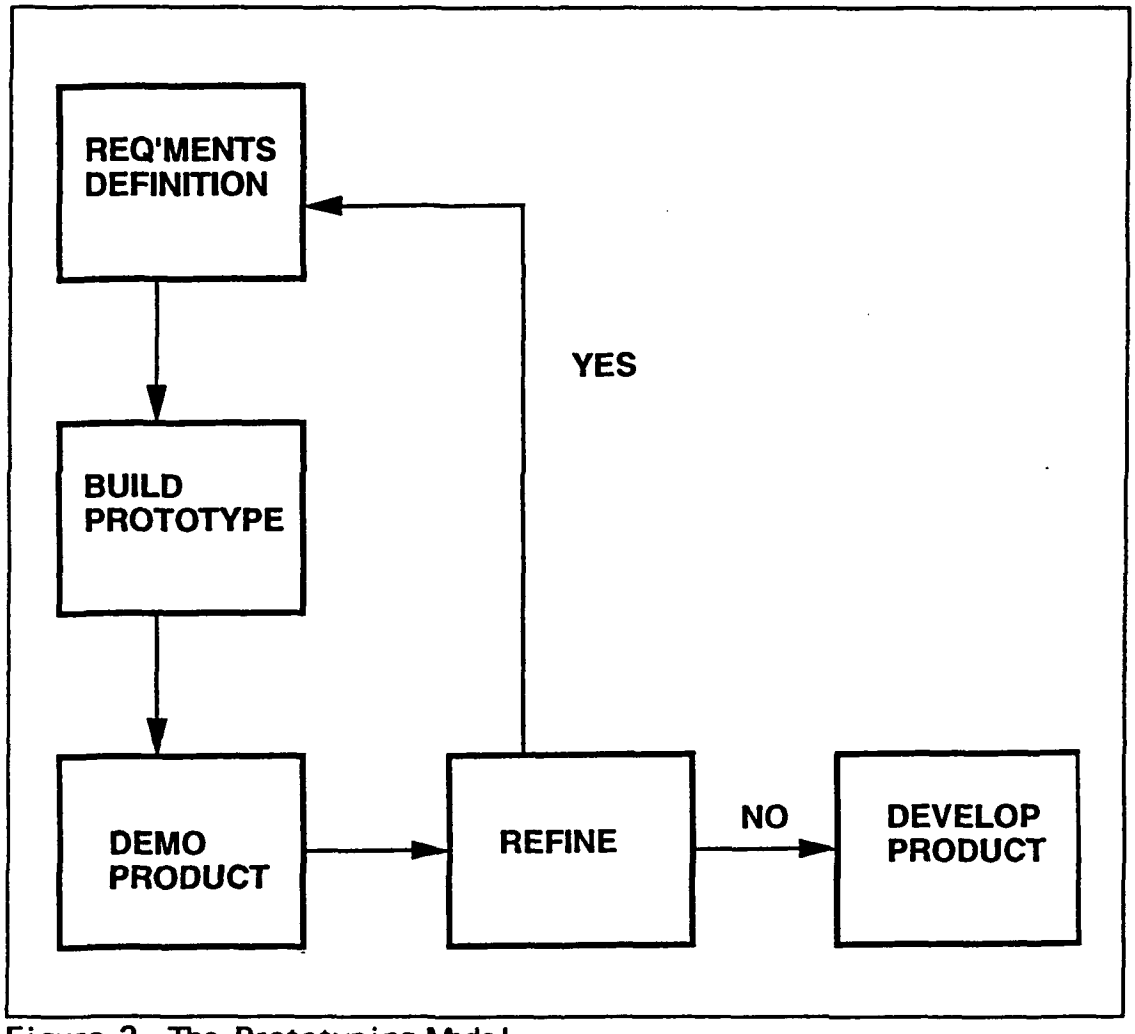

Figure 2 The Prototyping Model

There are three main activities within the software development process which may be influenced by the construction of prototypes: initiating the project, analyzing the business needs, and designing and constructing the software system.[Ref. 15:p. 3]

Various kinds of prototypes may be created. A *presentation prototype*, supporting the initiation of a software project, is used during acquisition to convince the client that

the future application system is indeed feasible. A *prototype proper*, a provisional operational software system constructed parallel to the information system model, illustrates specific aspects of the user interface or part of the functionality to clarify the problem in hand. A *breadboard prototype* is designed mainly for clarification of construction-related questions facing the developer team. Finally, if a prototype is used not only for experimental testing, but also in the application area itself as the care of the application system, it is known as a pilot system.[Ref. 15:p. 3]

One shortfall of the traditional software development methodology is the practice of waiting to perform extensive testing until one nears the project's end to ensure it fulfills all requirements. Thus, if a major fault is found at the end of the acquisition program, the anticipated lack of remaining project funds becomes a serious problem, as may be illustrated in today's characteristics of typically high software costs and low productivity. Alternatively, rapid prototyping bridges the developer to the user to agree upon a proposed system and to make continual assessments of its capabilities through an iterative process. Rapid prototyping may be an alternative to traditional software development methods or it may be used in conjunction with the traditional software development cycle.

A prototype does not need to implement all proposed product functions, although after requirements have stabilized, the design and structure of the prototype may be augmented to include those additional functions, even though the prototype may not meet all performance requirements. Thus, its structure may have to be transformed for

optimal performance and to account for differences between the host environment for the prototype and the operating environment for the proposed system.[Ref. 16]

E. COMPUTER-AIDED PROTOTYPING SYSTEM

Rapid prototyping provides the user with increasingly refined systems to test and provides the designer with increasing accuracy of feedback from the user, resulting in better engineered software. This is especially important for the case of hard real-time systems where inconsistencies are more likely. A real-time system requires more precision and accuracy compared to a conventional system, an example of which is the requirement that response times are met. The rapid prototyping system considered here is the Computer Aided Prototyping System (CAPS).

The main components of the CAPS (illustrated in Figure 3) are the prototype system description language (PSDL), the user interface, the software database system, and the execution support system. The PSDL is a high-level executable prototyping language designed to support the specification of real-time software systems and to organize and retrieve reusable components in the software base. It allows designers to sketch a system on a display using computer graphics and then refines the design with timing and control constraints in textual form. The user interface tools include the graphics editor, the syntax directed editor, the browser, and a system capability to generate English textual representations of PSDL specifications. The software database system consists of the design database, the software database, and the software design

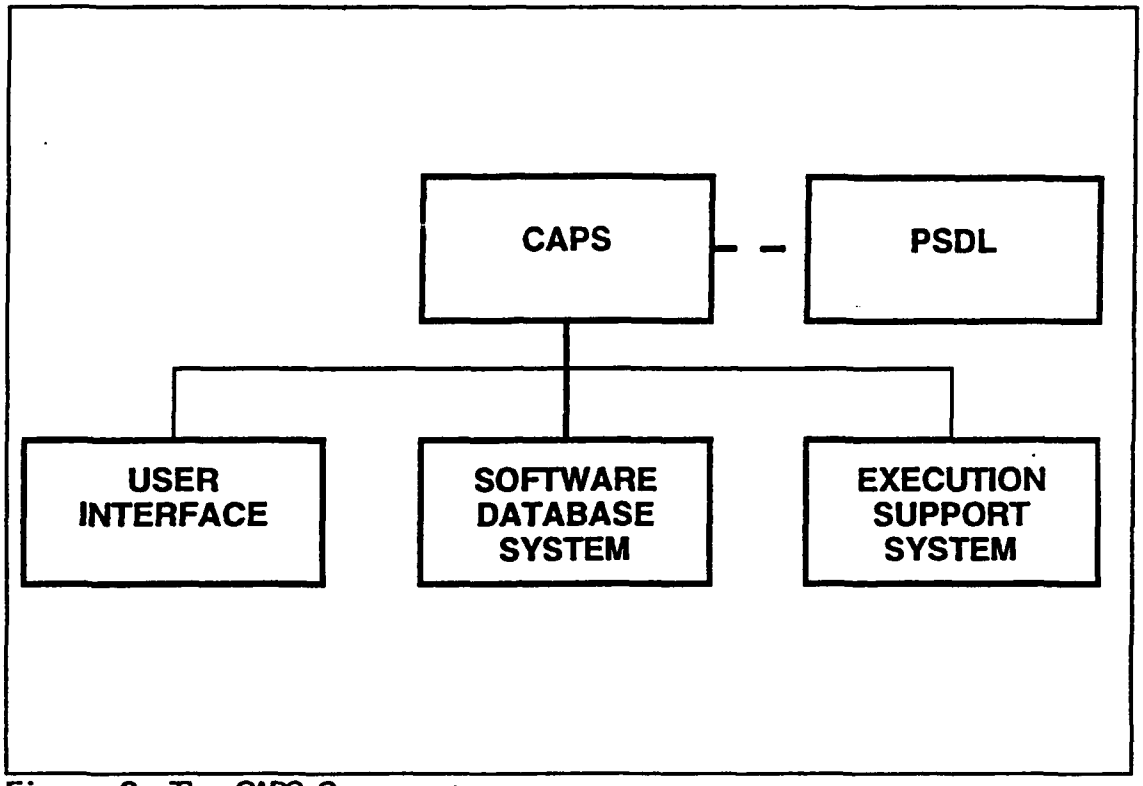

Figure 3 The CAPS Components

management system. The execution support system contains the translator, the static scheduler, the dynamic scheduler, and the debugger.[Ref. 17]

Facilities are provided through the CAPS for computer-aided design, software component reuse, and automated Ada code generation[Ref. 1]. These tools help software engineers construct and adapt software, validate and refine user requirements, and in checking the consistency of their proposed design. The concept of CAPS, as a prototyping tool, is to support the reaffirmation of software requirements as an iterative process between the customer and the designer by examining the executable prototype which is produced. The process supported by the CAPS provides both requirements and

prototype design in a form which may be utilized in the construction of an operational system. The four major stages within the CAPS process are: the software system design, construction, execution, and debugging (or modification). The requirements for a software system are expressed at different levels of abstraction and in different degrees of formality. The concept of CAPS is to provide the means to bridge the gap between customers and developers in meeting all levels. The CAPS has been designed to support quick prototyping through usage of a visual graphics (augmented data flow diagrams) mapped to a programming specification language which in turn generates executable Ada[1] code.

The CAPS creates a software prototype, that is, a mechanically processable and executable description of a simplified model of the proposed software system. It then modifies the model in an iterative fashion to refine the user's requirements. It therefore involves a two stage process of prototype construction and code generation. The first stage begins with the user defining the system's requirements from which the designer constructs the model. The user redefines these requirements when necessary. The process continues until the user attains satisfaction on the critical requirements being met. The second stage is then conducted and involves the transformation and augmentation of the model to code the final prototype.[Ref. 17]

Using the CAPS to engineer requirements conceptually provides several advantages over a manual determination. It offers a common baseline for users and software

---

[1]Ada is a registered trademark of the U.S. Government, Ada Joint Program Office

engineers. The defining of requirements within a domain specific language gives more efficiency and results in less error as well. It assists in the designer's proper interpretation of user requirements. Since the requirements for a software system involve moving from high to low levels, CAPS provides a useful representation of the information within a hierarchical goal structure with the customer's informal goals defined, then further refined at several levels with the natural language located at the highest levels and the prototyping language located at the most detailed levels. Finally, the CAPS may offer requirements traceability through the PSDL.[Ref. 17]

Concerning prototypes in general, a real-world implementation of the CAPS is one option which may be applied to ensure that the user has well-defined requirements, and the actual building of the real system will be implemented from a solid foundation. The prototype itself will serve no operational purpose; its usefulness will be in ensuring that the coding of the true product will yield a system that meets the user's validated needs. This correspondence may be maintained if user's needs change, resulting in evolving software requirements.

III. EVALUATION OF THE COMPUTER AIDED PROTOTYPING SYSTEM

A. OVERVIEW

This section provides a description of the various prototype demonstrations which have been created through application of the Computer Aided Prototyping System, the 1992 version. The individual prototypes addressed in this section are: the Patriot Missile, the Fish Farm, the Generic (C3I) Workstation, and the Sorting Tracks System. The Robot, also created, is a less complex prototype.

The rationale behind this description is to aid in providing an insight into how the CAPS could be applied to the acquisition cycle from the user's perspective through an understanding of what capabilities the CAPS may contribute, such as to an operations command. The following descriptions of each prototype is based upon personal observation from active sessions of viewing and experimenting with these prototypes.

B. GENERAL

The Computer Aided Prototyping System is the support software and hardware system which creates prototypes to help users define system requirements during acquisition, whether in the development or evolution phase of the program. The CAPS runs under a UNIX operating system. The 1992 version of the CAPS contains several demonstrations of prototypes which are addressed individually in the sections which follow this general discussion.

After starting the CAPS, the user enters the *Info* selection and chooses the *Design Database*. This presents the user with a main menu, from which there are various options, which are (in order of probable usage) Prototype Management, Graphs, PSDL, Ada Source, and Configuration Management.

1. Prototype Management

Before the user may access any of the offered capabilities within each prototype demonstration (other than *execute*), the database of each prototype must first be downloaded. This action is necessary because the user database initially contains just the executable files of the prototypes, and not the files which provide the other functional capabilities of the prototype demonstration. Also, at the end of each session, each database must be uploaded back into the main CAPS database. This action is necessary for this version of the CAPS and may become an automatic process in a later version of the CAPS, as it accounts for the assumption that the user will edit the downloaded database. Thus, if the user ends the session without first uploading, the current version of the CAPS will assume that the updated version was not returned to the main database and will not allow the user to access this database during his next session.

2. Execute

The execute option allows one to see how the various functions which are required to run work together, meeting the real-time constraints imposed upon them, and in turn proving that the requirements which are displayed in the data flow diagram and the textual description created through the PSDL is valid.

3. Graphs

For these existing prototypes, one conceptualizes that the user has worked with the designer to provide his requirements from which a data flow diagram has been created. This is viewed through the use of a graphics editor under the *Graphs* option. Using the graphic editor, the functions which the user wants carried out are depicted as circular nodes with streams (depicted as directed lines) for inputs and outputs to each functional node. This provides the user with an illustration or pictorial to provide a visualization of how his requirements are being implemented (Figure 4). Functional flow diagrams are an important part of the requirements portion of acquisition because the interaction (or dependencies) between functional nodes is thus realized. There are several options provided within the graphics editor. Of these, *select*, *specify*, *streams*, and *constraints* are of most interest to the user. The graphics editor has the options of creating each functional node (or bubble) and each data stream (or link), with the flexibility of various color, pattern, font, and other edit options.

The CAPS provides the capability for the user to click onto any functional node using the *specify* command to bring up a window of PSDL text; that is, the specification for that function. This text identifies the inputs and outputs of the function, as well as the inclusion of a description of the function. For the PSDL of a data stream, the user would select the stream and click onto it for a textual description pertaining to that data flow.

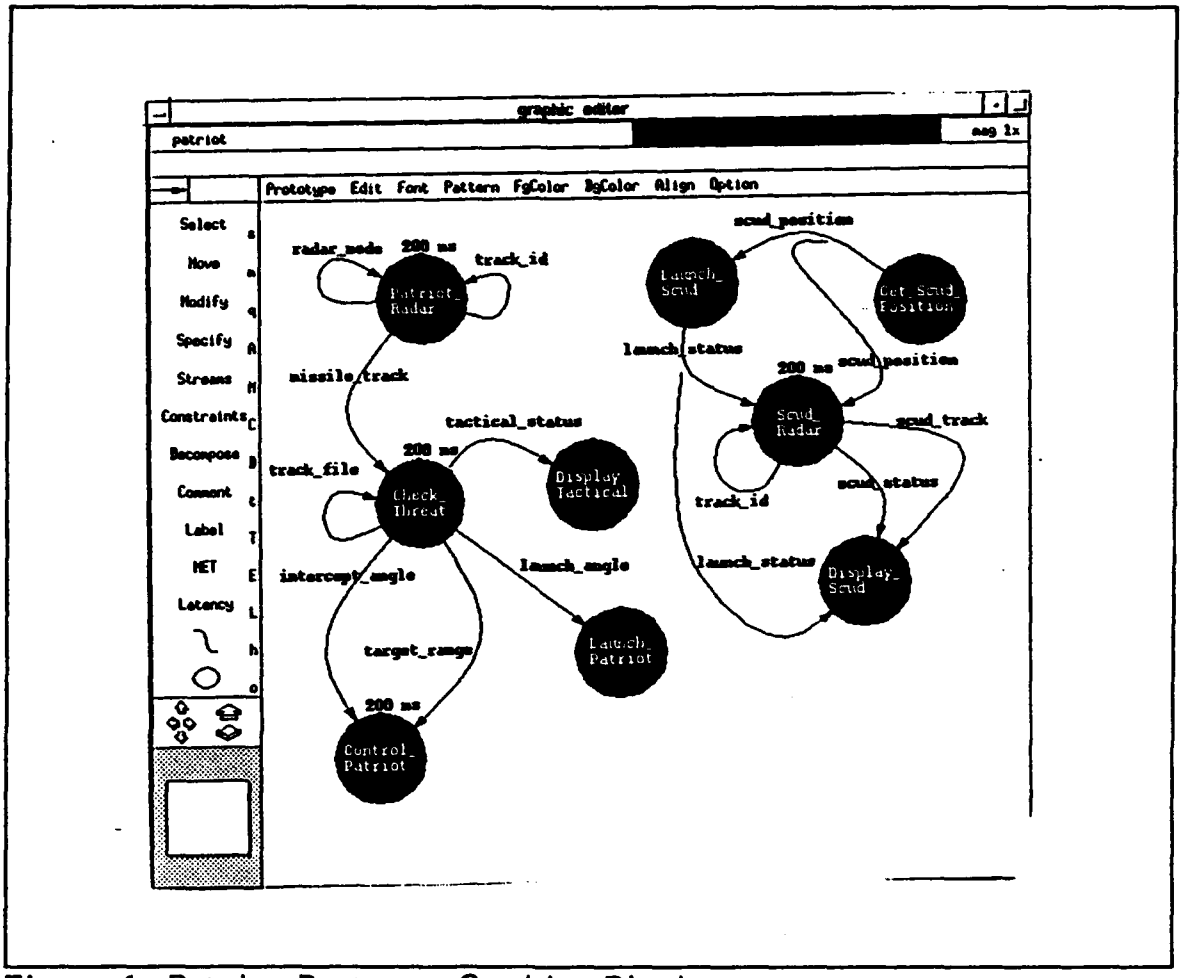

Figure 4 Patriot Prototype Graphics Display

The CAPS also provides the capability for the user to click onto any functional node using the *source code* command to bring up a window of the Ada source code corresponding to that function.

For the source code corresponding to a data stream, the user would select the stream and click onto it to view the code corresponding to that data flow.

The CAPS automatically generates the shell of the PSDL as well as the limited addition of further code, such as for timing constraints. The remaining code pertains to the actions of the functions themselves and is usually prepared manually by the designer, unless a canned module already exists within the software database from which the designer may draw upon.

After the required number of iterations, or when the user is satisfied with the prototype, the process of acquisition may continue to its next stages involving the creation of the true end product. The probability of successful fruition of user requirements within the final software product will be greatly enhanced by the refinement of those requirements through the discipline of rapid prototyping.

4. Prototype System Description Language (PSDL)

The prototype system description language is the textual specification which corresponds to the functional flow diagram of the prototype. A user may view a prototype's PSDL through two methods. The first method was previously addressed in the *Graphs* section. The second method for the user is to choose the *PSDL* option from the menu initially presented, after specifying the prototype desired. The user may then view the text in its entirety without going to the functional flow diagram. However, if the user wishes to see only the textual specification which corresponds to a particular function, then the user would first access the flow diagram as previously described in the *Graphs* section.

5. Ada Source Code

The Ada Source Code, which is the actual textual programming language for the prototype, may be accessed through two methods. The first method is explained in the *Graphs* section. The second method is for the user to choose the *Ada Source* option from the menu initially presented, after specifying the prototype desired. The user may then view the source code in its entirety without going to the functional flow diagram. However, if the user wishes to see only the specific source code which corresponds to a particular function, then the user would first access the flow diagram as previously described in the *Graphs* section.

The above text has provided a discussion of CAPS on a general level. The following sections will now describe the individual prototype demonstrations which were created through the CAPS.

C. THE GENERIC COMMAND, CONTROL, COMMUNICATIONS, AND INTELLIGENCE (C3I) WORKSTATION

The Generic C3I Workstation (Figure 5) was designed for various platforms to support the command and control architecture of a composite warfare commander to provide support in monitoring the air, surface, subsurface, and power-projection tactical environments to aid in the commander's decisions. This prototype demonstration emulates a command, control, communications, and intelligence (C3I) system. It evolved as a thesis project for naval students and was designed to accommodate a large number of tracks, integrate dissimilar source information and provide a commander with timely

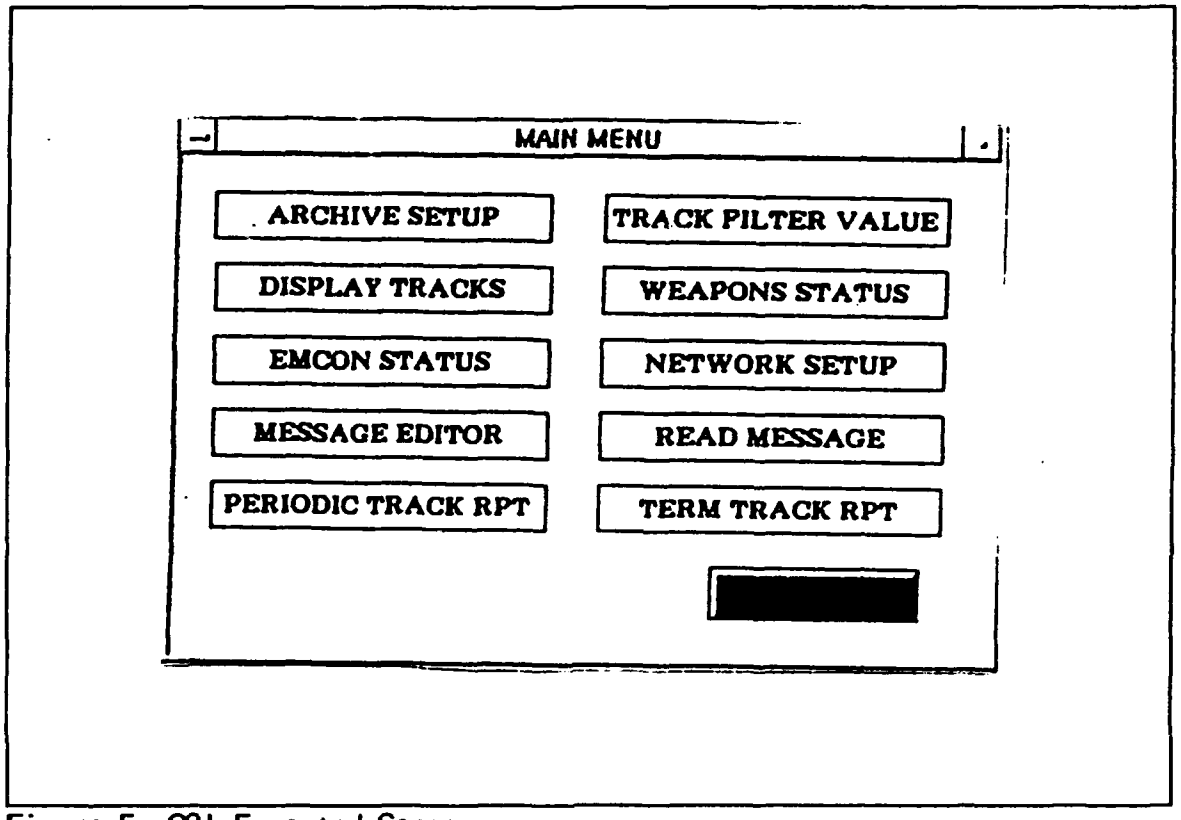

Figure 5 C3I Executed Screen

tactical information[Ref. 18:p. 4]. The initial planning for this project was to develop the prototype and then to utilize it in an operational environment with an input of real data. This is presently the most complex of the prototypes created with the CAPS. It is characterized by an open system architecture which allows modifications to a portion without unduly impacting the system as a whole. It is capable of displaying both graphical and textual views of the current situation within a geographical area, showing the most recent status of track information that has been provided from own-ship platform sensor inputs, communication sources, and manual inputs. It also provides

own-ship states of weaponry for usage in performing a battle damage assessment or in generation of situation reports[Ref. 18]. To analyze the prototype, one may first perform the execution of the code. Once the system is executed, its main menu provides the options for: an archive setup, a track filter value, the display tracks, the EMCON status, the message editor, a periodic track report, a term track report, the read message, the network setup, and a weapons status. Within the EMCON status, the options given are *silence* and *no silence*. Within the Network Setup, the options of JTIDS, Link11, Link16, and OTCIXS are provided. Within the Archive Setup, one identifies whether this setup pertains to *all ships* or *own ship* as well as the choice of the same options as under the Network Setup. The Track Display shows various IFF classes (friendly, hostile, neutral, unknown), as well as the track classes of air, surface, and subsurface. Also included are the range and message arrival. Its tracking-information includes such items as the latitude, longitude, speed, range, and course. The Weapons Status identifies such status as *ready* or *reloading* for MK48, CIWS, GUN, and TWS. Finally, the Track Filter provides the track classification as well as displays the maximum number of tracks.

Using the graphics editor to display the functional flow diagram for C3I, the functional nodes identified are: communications links, communications interface, track database manager, user interface, navigation system, sensor interface, sensors, weapons system, and weapons interface. An example of a constraint is communications links, which is *Operator -- Period 30000 Ms*. As an aside, no matter what node was clicked upon, the same set of constraints were displayed; that is, it was all-inclusive. For

example, the viewing under the communications links node also contained the constraint for the weapons interface. This will be corrected in the 1993 version of CAPS.

Using the graphics editor option of *specify* for communications links, its PSDL specification was displayed. As mentioned in the section of general application, one could view PSDL for each function (or bubble) or stream, displaying specification text for each.

D. FISH FARM

The Fish Farm (Figure 6), a short-term prototype in comparison to the Generic C3I Workstation, was developed to be an example of an embedded real-time control system[Ref. 19:p. 2]. Originating from a class project, this prototype is a demonstration of the CAPS as a tool for providing software methodology and design environment, a short-term prototype in comparison to C3I. Fish Farm contains three separate hardware devices in its control system: sensors, valves, and a feeder. It controls both a fish food dispenser and the quality of a fish pond's water content. The fish food is delivered at scheduled feeding times, daily. To carry out this function, a mechanical feeder drops pellets of fish foods into the pond and is switched on or off by the computer. The computer also controls the water inlet pipe and drain pipe valves. Its sensors measure the water, oxygen, and ammonia levels of the water. One of its most notable features is its use of TAE (a government-owned software package, which is a category of *public domain software*) to provide a pictorial environment. Thus, the Fish Farm is a good example of interfacing with the control system of CAPS. This prototype's executable

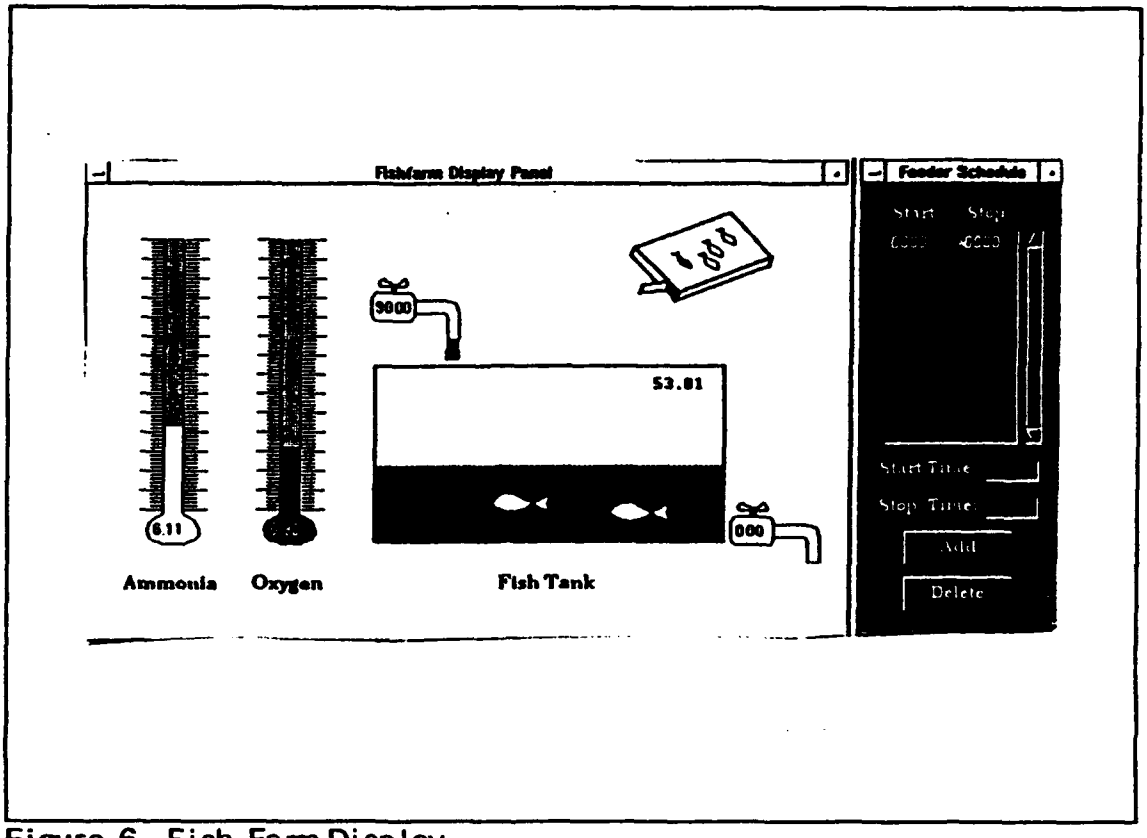

Figure 6 Fish Farm Display

brought up a fish farm display panel to depict a fish tank with concentration of ammonia and oxygen levels provided. Also part of the illustration is a feeder schedule. Within the graphics editor, the functional nodes were identified as monitors of oxygen, NH3, and water; and, control for water level and feeding.

E. PATRIOT

This prototype, like Fish Farm, is a class project which utilizes CAPS as a tool for software methodology and design environment. It also makes use of the TAE for detailed graphics (Figure 7). The execute function of the prototype brought up the

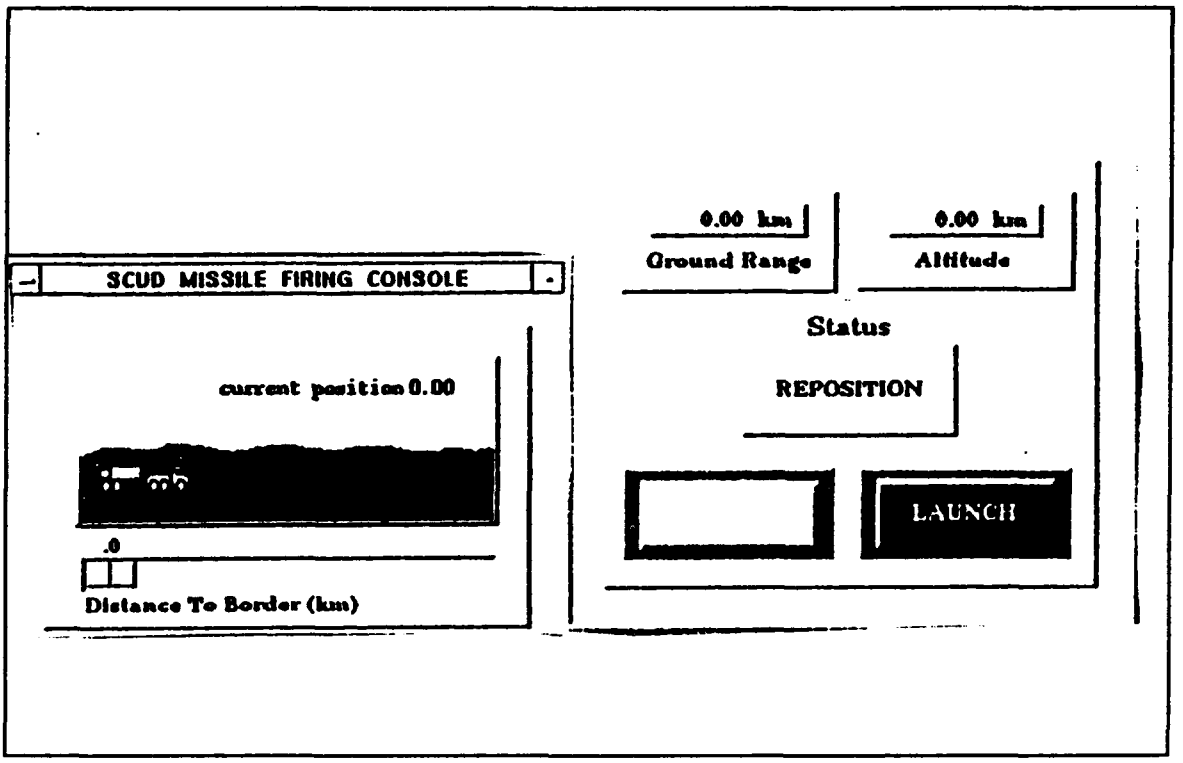

Figure 7 Patriot Pictorial Display

patriot missile defense system (Figure 8). This prototype depicts a Patriot Missile Defense System for simulation of a patriot missile intercepting a scud that has been launched. When the user executes the prototype, the following windows are presented to the user to simulate status charts: Radar Status (with a *search* option), Threat Status (with altitude provided as *predicted*, *impact*, *time* in seconds, and *impact point* in kilometers, as well as *distance from patriot emplacement*), Patriot Status (with *predicted*, *intercept time* in seconds, *time of flight* in seconds, and *distance from patriot emplacement* provided), and a Scud Firing Console (with *current position, distance to border, ground range* in kilometers, and *altitude* in kilometers, as well as the options of

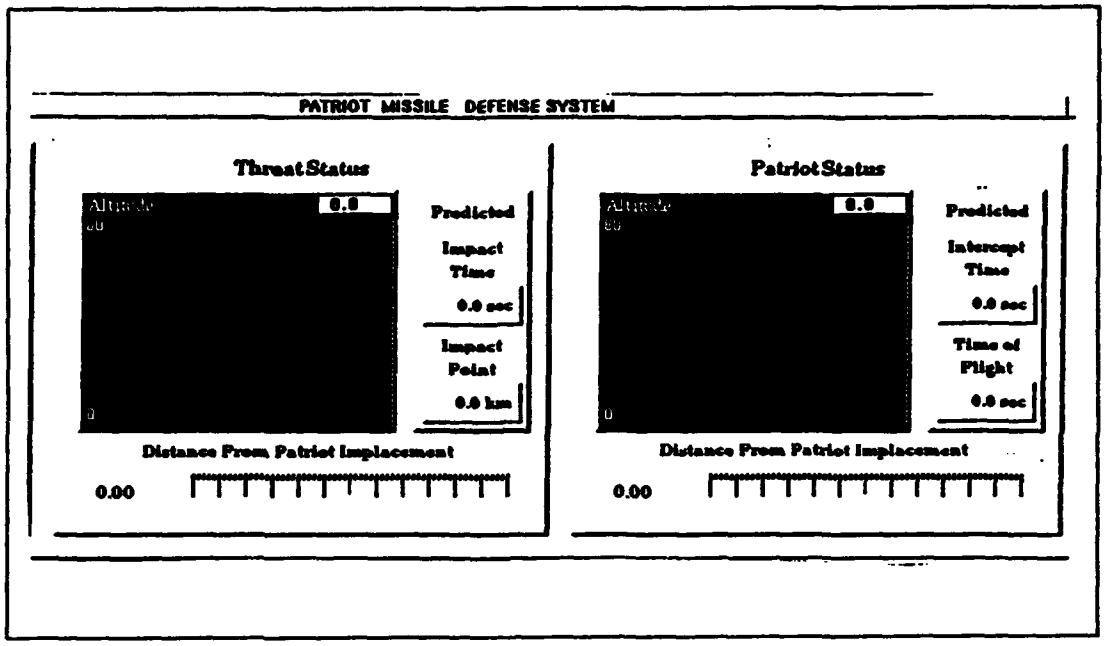

Figure 8 Patriot Executed Display

status, reposition, and reload launch included). Within the graphics editor, the following nodes are displayed (refer back to Figure 3) for the patriot system: Patriot Radar (with inputs of Radar Mode and Track Identification, and output of Missile Track), Check Threat (with inputs of Missile Track and Track File, and outputs of Tactical Status, Launch Angle, Intercept Angle, and Target Range), Display Tactical (with input of Tactical Status), and Launch Patriot (with input of Launch Angle), and Control Patriot (with inputs of Target Range and Intercept Angle). The following nodes illustrate the scud firing system: Launch Scud (with input of Scud Position and output of Launch Status), Get Scud Position (with output of Scud Position), Scud Radar (with inputs of Launch Status, Track Identification, and Scud Position, and outputs of Scud Status and

Scud Tracking), and Display Scud (with inputs of Launch Status, Scud Status, and Scud Tracking). The prototype's screen pictorial of this flow diagram created by the graphics editor is shown in figure 1. The layout of the nodes and their data streams is a typical illustration of what the graphics editor is capable of creating. The actual prototype pictorial is enhanced by color; the patriot nodes are in blue while the scud nodes are in red.

F. TRACKS

The notable feature of Tracks is that this very simplistic demonstration of few modules covers the full process of CAPS capabilities, including the illustration of retrieving re-usable software (in this case, a sorting program). This is still one of CAPS's capabilities which is in a very early stage of implementation. This prototype demonstrated the tracking of approaching elements and their probable identification. The execution displayed the various elements tracked: azimuth, range, and heading speed. The identification function was also displayed to describe a tracked element as: friendly, hostile, or unknown. Within the graphics editor, the following functions were displayed: *read, sort, and write* tracks.

IV. AIR FORCE/DOD SOFTWARE ACQUISITION CYCLE

A. MAJOR OBSTACLES WITHIN SOFTWARE ACQUISITION

This section discusses the major obstacles that program managers commonly face within the software acquisition cycle. These obstacles are separated into the areas of cost, productivity, maintenance, efficiency, portability, security, reliability, quality, delivery schedule, real-time requirements, and a general category of other considerations.

1. Cost

Cost is perhaps the largest obstacle within a software acquisition program, and both direct costs and indirect costs are normally considered. The direct cost of software is the price to be paid for the processing of designing and coding the product, while indirect costs are additional costs which result from the adverse affects of software delays and software errors, such as impacts upon the operational readiness of a system, the sortie rates achieved by an aircraft, and other operationally oriented concerns. This in turn could lead to significant dollar losses within the impacted service. There are many factors that influence software costs, some of which are: personnel, management, complexity, structuring of programs, automated aids, hardware, and interaction. In one survey that was conducted, the distribution of Air Force software costs, organized by application, were reported as: management information systems (33%), scientific and engineering (23%), command and control and intelligence (21%), logistics and maintenance (13%); and avionics (10%)[Ref. 20:p. 29].

The cost incurred for software acquisition cannot be focussed upon software development costs alone, but must more accurately consider software life cycle costs in total. Life cycle cost problems include:

1. high initial development cost
2. high operations and maintenance cost
3. costly modifications
4. high cost of documentation, and
5. poor modifications.[Ref. 20:p. 29]

The principal sources for life cycle cost problems are:

1. poor estimation of production costs and schedules
2. poor procurement
3. poor software development practices
4. Lack of automated programming techniques
5. Improper use of existing developments or lack of use
6. Inadequate system hardware
7. Inadequate programmer skills levels
8. Poor system requirements and specifications
9. Lack of management control of costs
10. High salaries of programmers
11. Uncertainty of cost allocation

12. Inadequate attention to system integration and testing, and

13. Poor documentation practices.[Ref. 20:p. 48]

2. Methodology

Another area for improvement is the lack of management's usage of practical and effective measurements for adequate control within software development. As DoD systems continue to grow both in complexity and cost, there is a valid concern for greater measurement and control within the software acquisition process. This becomes an even greater issue when dealing with complex systems, especially mission critical systems, where achieving superior software quality is of such emphatic importance. There are still shortfalls in the attempt to attain these goals of measurement and control. The failure to achieve this has been due in part to a lack of understanding toward development and maintenance of software with engineering as its underlying philosophy. That is, the application to software of the traditional engineering process rather than alternative approach of treating software as a work of art. The traditional engineering process considers the following activities:

1. The iteration between formal analysis and design,
2. The utilization of earlier design,
3. The alternatives and their tradeoffs,
4. Manuals or Handbooks,

5. A good cost-effectiveness approach, and

6. Concentration on economic concerns.[Ref. 21:p. 2-3]

The software engineering approach pertains to the military more than the approach of viewing software as an art. The latter is found to be more of an individually-crafted and personality-intensive software construction process. A characteristic of military systems is of being too large to develop cost-effectively using this hand-tooled "software as art" model. Along these lines, it is also of importance to note that the commercial sector concentrates more on engineering and less on documentation than does the government. This allows software to be somewhat independent of its textual references and eliminates the large requirement for documentation. This concept could be reflected in updates to the Department of Defense System Software Development, Mil-Std-2167A. Another consideration is that a disciplined software engineering practice enables managers to control the process of software development and provides software engineers with a foundation for building high quality software.[Ref. 21:p. 2-3]

Even though software measurement has not reached a stage of maturity that allows consistent interpretation of the result, there is a direct relationship between measurement and managing or improving software. Software measurement can help bridge the gap between management and the technical staff. The lack of attention to metrics can be cited as follows:

1. A core set of software metric charts should be briefed at each Air Force Review.

2. The OSD and services haven not defined a standard set of core software management metrics for use on all programs.

3. Program contract offices need guidance on what software -related data should be placed on contract as deliverable to support software management metrics.

4. PMs do not understand software and need training on how to interpret development test results and software metrics.

5. DT does not evaluate software maintainability.

6. Government and contractors have software standards but due to lack of training they are not followed.

7. Work breakdown structures do not adequately address software development and testing.[Ref. 21:p. 2-3]

Organizations with experience in quality problems may not have a good system of quality measurement, and this lack of measurement normally maps to inefficient management. Thus, there is a push for standard software measurement in the Air Force to ensure visibility into the software acquisition and development process and the products themselves.[Ref. 21:p. 2-3]

3. Software Risks

Software is considered to be one of the most risk-prone of all engineering activities. Some of the more common risks are schedule slips and cost overruns, which tend to occur on more than half of all large systems. Other less common, but still severe risks include the cancellation of a project prior to its completion or that are completed

with serious quality deficiencies. Something which aids in this area is the field of software risk control, which is expanding rapidly, although many projects still lack formal risk management approaches. Risk identification and risk avoidance often depend informally upon the skills and experience levels of software managers. The magnitude and severity of risks that are not caught by informal controls supports the conclusion that management experience, by itself, is not a sufficient safeguard against software risks.[Ref. 22:p. 17]

4. Productivity

One of the most visible problems facing the systems development profession today is insufficient productivity of the systems designer and programmer. The quality of work performed by systems analysts may also have notable impact on productivity. A characteristic which reveals this problem to varying degrees is that of a backlog. Types which are included in the definition of a backlog are:

1. Visible backlog -- new systems requested that have been approved and funded but not yet begun due to inadequate resources
2. Invisible backlog -- systems that users want but have not asked through official channels because of the visible backlog, and
3. Unknown backlog -- systems that users do not even know they want but will be identified when the visible or invisible backlogs are completed.

A second aspect of the productivity problem is the length of time required to develop any individual system. Yet another issue is those projects which result in failures and are then cancelled prior to their completion. As many as 25% of all

projects in large MIS organizations are never completed, as suggested by various surveys[Ref 23: p. 107].

A final concern is that much of the work of developing an automated information system may be carried out in a manual fashion. To attempt to resolve these problems, techniques such as increasing the workforce of programmers and systems analysts, hiring more highly-skilled personnel, and allowing users the leverage to develop their own systems are options which are considered. Another method to attain increased productivity is improvement of programming languages. Also, another approach is in the area of software engineering disciplines. These are a collection of tools, techniques, and disciplines for the support of software development and include such practices as structured programming, structured design, structured analysis, software metrics, and software quality assurance.[Ref. 23]

5. Maintenance

Maintenance is a major issue in the systems development field as well. This may be addressed from two viewpoints: maintenance of newly developed systems and continued maintenance of older systems. The correction of ongoing errors is one aspect of maintenance. It accounts for approximately 21% of the overall maintenance effort in American data processing organizations[Ref. 23:p. 114]. Maintenance also involves modification of a system to reflect changes in the hardware modifications to speed up certain operational aspects of the system or modifications to reflect a change in the end user's requirements of the system. Software maintenance is a major problem for most organizations; between 50% and 80% of the work done in most systems development

organizations is associated with the revision, modification, conversion, enhancement, or debugging of a computer program that someone else wrote. Maintenance is expensive. In the early 1970's, the DoD reported the cost of developing computer programs on one project average $75 per computer instruction but the cost of maintenance of the system ran as high as $4000 per instruction.[Ref. 23]

6. Efficiency

A system not operating with an appropriate throughput and with an acceptable response time for on-line terminals is yet another obstacle to the software acquisition cycle.[Ref. 23]

7. Portability

Most new systems are implemented on one brand of computer, but a problem is encountered if there is failure to develop the software such that it may be moved to different computers with ease.[Ref. 21]

8. Security

Since modern computer systems are highly accessible and are responsible for ever increasing sensitive information, security is a major issue for many development projects.[Ref. 23]

9. Reliability

Failure to meet acceptable levels of reliability is another obstacle to software acquisition. On average, software developed in American organizations has between three and five errors for every hundred program statements -- after the software has been

tested and delivered to the customer[Ref. 23:p. 112]. Other reports suggest American software may have as many as three to five error for every ten program statements[Ref. 22]. Software errors range from the trivial to major. Some errors are never found and the process of documenting and recording errors is so inadequate that half of errors found are not reported, one survey suggests[Ref. 23:p. 113].

10. Quality

Failure to reach acceptable levels of product quality is a major obstacle in software acquisition. Characteristics for poor quality of software include: unreliability, unresponsiveness, incompatibility, nonadaptability, nontransferability, and uncertifiability. The principle causes of software quality problems are:

1. Inadequate statement of requirements by user
2. Inadequate understanding of user requirements
3. Poor testing and certification practices
4. Lack of standards by which performance can be measured
5. Inadequate documentation
6. Lack of appropriate management attention and control
7. Improper use of current technology
8. Inadequate programmer skill levels
9. Inadequate hardware or software trade-offs, and
10. Lack of adequate support software.[Ref. 20]

11. Delivery Schedule

The following problems occur in regard to delivery schedule: failure to meet schedule, long development time, and untimely software documentation. The principal causes for these problems to occur are: poor estimation practices, inadequate definition or understanding of the job, variable programmer skills and productivity, poor management control and monitoring, unrealistic milestones, inadequate use of existing developments, long lead-time procurement, inadequate support software, lack of automated programming activities, and inadequate attention to documentation.[Ref. 20]

12. Real-time Systems Requirements

Errors made early in a system design are discovered late and are the most difficult and expensive to correct. In particular, if an error is made in identifying and recording the requirements of a project, it is unlikely to be found until the completed system is subjected to field test, at which time the consequences of the mistake are likely to pervade the whole of the system design and correcting it may involve extensive reworking of the complete system. The difficulties for large-scale real-time embedded systems are exacerbated by aspects of requirements which involve time. There are temporal aspects of performance to be considered; there are events that must take place at fixed times in real time, within some tolerance; there are safety issues that involve defining circumstances in which urgent corrective actions must occur within tight limits of time, or times when events must not take place; there are concurrent actions by different elements in the system which must pursue their separate existences, disturbed only by interactions with other concurrently active system elements. One of the great

weaknesses of most existing formal schemes in dealing with time-critical systems has been the problem of dealing with time itself. First, there is frequent assumption that transitions from one state (of the system) to another are instantaneous. The semantics of such systems are then explained in terms of transition graphs and trees, both potentially infinite. Many classes of application cannot be described easily or at all if one keeps the assumption instantaneous of action. A richer logic is therefore required.[Ref. 24]

13. Other Considerations

The following segments discuss miscellaneous areas which cause difficulty in the arena of software acquisition.

a. Inconsistencies

One of the causes of uncertainty in software development is the lack of meaningful standards through inconsistencies within the guidance and inadequate performance measures.[Ref. 20]

b. Standard Language

The consideration for a standard language is meaningful. There are so many different languages available (these languages even differ from machine to machine) and many versions within these languages that the lack of language standardization may be appreciated. There is a direct relationship between lack of standardization of languages and software cost measured in terms of the price of additional documentation; nontransferability of code, compilers, and the like;

development of new tools; unreliability; and the resultant large software inventory that is required. Standard operating system interfaces are needed to achieve reduction in errors over that obtainable with current job control languages. A standard data definition language which eliminates the need for many data structures is another important requirement.[Ref. 20]

c. Repetitious Factors

The repetitious presence of such factors as insufficient requirements, inadequate attention to testing, documentation, and integration, poor software management, lack of support software, and utilization of outdated techniques and tools points to these as primary problems within the software community.[Ref. 20]

d. Software Types

One must also consider the differences between real-time software and batch-scientific or business-type software. The latter requires little testing because the programs are typically similar and errors are more likely to deal with formatting and such, and are easily fixed. Real-time software, however, involve timing errors which are more difficult to deal with[Ref. 20]. Real-time software must execute multiple processes effectively within set timing constraints and provide mechanisms for svnchronous and asynchronous process communications. Typically, embedded in complex systems, its correctness depends upon the time at which the results are produced as well as the logical results of computation. Thus, real-time systems may fail if the system cannot execute its critical workload in time. Problems to be addressed during

their software life cycle include: specification and verification techniques, process scheduling, communication architectures, and automated, systematic hard real-time software development methods.[Ref. 19]

e. Requirements Specification

Contemporary development methods recognize three potential problem areas in the complex task of requirements specification. First, an analyst's reasoning process must be guided by an underlying process which is appropriate to the task as well as the problem domain. It must be generic in nature and represent a standard approach within an organization so that a specification generated by a development team is achieved in an integrated way. Second, facilities must be provided for locating information about an evolving specification. Facts gathered during the process of constructing a requirements specification must be correlated, irrelevant ones discarded, and appropriate facts organized in meaningful structures. Third, assistance is needed in the communication between analysis and end-users during the phases of facts acquisition and specification verification. Capturing and verifying requirements are labor-intensive activities which demand skillful interchange between those who understand the problem domain and those that need to model the problem domain.[Ref. 25]

f. Quality

In software engineering, there seems to be no well developed sense of quality. There are generally no means to determine the current state and compare it

with the earlier state of the system or the future desired state. This precludes tracking of progress and inability to plan the future.[Ref. 20]

g. Large Systems

Large real-time systems are today often based on huge software systems. The rapid development on the hardware side has not been matched by rapid development on the software side, although the latter may dominate in terms of development efforts. As software is becoming a greater and more important part of a system, it is necessary to incorporate metrics and models for software performance analysis together with standard performance measurements for a more complete and comprehensive modelling of a total system.[Ref. 23]

h. Software Requirements Documentation

Statements of user requirements, in the traditional sense, can be far too extensive to be feasible; functional specifications normally must be read in their entirety in order to glean an understanding. Quite often, sections within specifications are found to be highly redundant, and unnecessarily so. That is, the same information is often repeated in several different parts of the document.[Ref. 23]

i. Tracking Requirements

One of the major sources in contributing to software cost is that of tracking user requirements as they change during the development phase. Also, hardware costs are to be considered as they may result from manufacturers' hardware-

related software costs. Finally, documentation costs must also be accounted; these are normally higher for business-type programs.

j. Miscellaneous Findings

Other findings include as examples of obstacles in acquisition management:

1. The lack of visibility of the development process.
2. The difficulty of applying DoD-Std-2167A to prototype and incremental build developments.
3. The need for contractual mechanisms for changing requirements.
4. Contractors do not follow their own guidelines and methodologies.
5. There are not enough resources (dollars and people) to do the proper job.
6. Engineers are inexperienced and need to be kept abreast on software technology and the use of the DoD standards.[Ref. 26:p. 4]

B. PORTION OF THE SOFTWARE DEVELOPMENT LIFE CYCLE WITH MOST PROBLEMS

This section will discuss the key portion of the software development life cycle (from the classical project life cycle view) which typically suffers the most setback, that of the late stage of a program when formal testing is conducted.

1. Classical Project Life Cycle For Software Development

The use of bottom-up implementation is one of the major weaknesses in the classical project life cycle. The programmers are expected to carry out all their module

testing first, then subsystem testing, and finally system testing. This approach is also known in the computer industry as the Waterfall Life Cycle.

2. The Independent Validation And Verification Stage

The bottom-up implementation approach has a number of serious difficulties which lead to impacts in this stage of the software development life cycle, as addressed in the following segments.[Ref. 23]

a. Inflexibility

A major weakness with the classical project life cycle is its insistence that the phases proceed sequentially from one to the next. This can prove to be a highly inflexible approach not allowing for real-world phenomena which may impact the program.

b. Schedule

Nothing is completed until it is all finished. Thus, if the project gets behind schedule and the deadline falls right in the middle of system testing, there will be nothing to show the user as a finished product.

c. Serious Errors

The most trivial bugs are found at the beginning of the testing period and the most serious bugs are found last. Major interface error found at the end of a development project can lead to the recoding of large numbers of modules and can have devastating impact on the schedule.

d. Debugging

Debugging tends to be extremely difficult during the final stages of system testing. Debugging is the process of discovering where the bug is located and subsequently how to fix the bug after the process of testing has determined that there is a bug. It is often extremely difficult to tell which module contains the bug if it has been discovered during system-testing of a bottom-up project.

e. System Testing

The requirement for computer test time usually rises exponentially during the final stages of testing. More specifically, the project manger often finds that he needs large contiguous chunks of computer time for system testing per day. If the computer testing time cannot be obtained, the project falls behind schedule, sometimes seriously.

C. WHERE CAPS COMES INTO PLAY, WHERE IT COULD BE USED, AND BY WHOM

A brief mention of prototyping in general will be provided followed by a discussion of The Computer-Aided Prototyping System (CAPS) as it relates to the software acquisition life cycle. The portion of the life cycle that it most comes into play, where it would be utilized within that arena, and the personnel who would make use of the system will be covered in this section.

1. Prototyping

A variation on the top-down approach is that of prototyping. It is defined by Yourdon, Edward:

> An alternative approach to requirements definition is to capture an initial set of needs and to implement quickly those needs with the stated intent of iteratively expanding and refining them as mutual user/developer understanding of the system grows. Definition of the system occurs through gradual and evolutionary discovery as opposed to omniscient foresight... This kind of approach is called prototyping. It is also referred to as system modeling or heuristic development. It offers an attractive and workable alternative to prespecification methods to deal better with uncertainty, ambiguity, and fickleness of real-world projects. [Ref. 23:p. 97]

The prototyping approach assumes that the system will be modelled by a working model, i.e., a collection of computer programs that will simulate some or all of the functions that the user wants. Because those computer programs are intended as just a model, it is assumed that the model will be discarded eventually and replaced with the true product.[Ref. 27]

There are many benefits which can be realized through prototyping. The prototyping approach is better for sudden or major changes. Thus, at the event of freezing the project, the likelihood of having a system prepared for demonstration is greater than a non-prototyped system.[Ref. 27] This approach provides a superior environment for knowledge elicitation through allowing the expert to criticize working models of the final system.[Ref. 27] This approach allows for greater flexibility in project planning.[Ref. 27] Testing in the prototyping approach is spread out through the project whereas in the non-prototyped system, the testing is held at the very end of the

project.[Ref. 25] With the latter method, finding and correcting deficiencies can be a much more difficult and costly process.

Good candidates for a prototyping approach are projects with the following characteristics:

1. The user is unable to examine abstract paper models such as data flow diagrams.
2. The user is unable to articulate his requirements in any formal way and can only determine the requirements through the process of trial and error.
3. The system is intended to be on-line with full-screen terminal activities, as opposed to batch edit, update, and report systems.
4. The system does not require specification of large amounts of algorithmic detail, i.e., the writing of many process specifications to describe the algorithms by which output results are created.[Ref. 23]

2. The CAPS within Software Development

The software development process includes the following major activities:

1. System Requirements Analysis/Design
2. Software Requirements Analysis
3. Preliminary Design
4. Detailed Design
5. Coding and CSU Testing
6. CSC Integration and Testing
7. CSCI Testing
8. System Integration and Testing[Ref. 4]

CAPS involvement would be heaviest in the early stages of the software acquisition process, that is, during requirements identification and analysis by: determining initial user requirements, building the prototype, demonstrating the prototype and adjusting requirements through iterative processing. Because major modifications to systems already late in the design process (or production) may prove costly, requirements analysis is a stage of the software development process which should not be overlooked. A CAPS similar to the present research would not be involved in the later stages of software acquisition that involve the production model, the true product. Beginning with detailed design and on into testing of the product, CAPS would already have performed its role in aiding user requirements definition for an accurate flowdown into product code.

A further reaching concept of CAPS as a tool during both the requirements analysis and software development would carry the above a step further by implementing the system while continuing its optimization.[Ref. 19] The current research model of CAPS is much closer to real-world application within the requirements definition process, but much further enhancement to the system is needed before its application to coding a production system. But given this ability, the process of software development using CAPS could then consist of the following steps:

1. Determine initial requirements
2. Construct prototype

a. Find reusable components

b. Decompose

c. Write Ada code

3. Demonstrate

a. Generate schedule and executable

b. Demo typical scenarios

c. Get feedback

4. Adjust requirements and iterate

5. When stable, implement and optimize

a. Complete non-critical parts

b. Transform to gain efficiency

c. Port to operating environment[Ref. 19:p. 13]

A CAPS that meets the intent of its present prototype system would aid in decreasing the software's time of development and consequently, the cost as well. Another application could be its use after the system enters production or deployment to assist analysis of impacts from changing requirements.

V. THE APPLICATION OF COMPUTER AIDED PROTOTYPING SYSTEM, A CASE STUDY

A. OVERVIEW

This section presents an evaluation of the following two methods toward creating a system prototype: the 2167A (or classical waterfall) process in which the software is coded manually, and the process of rapid prototyping where part of the code is automatically generated via computer aid, in this case through the CAPS. The evaluation will cover the phases of requirements analysis and feasibility study for the project life cycle.

The Generic C3I Workstation will be the model for the evaluation. This CAPS prototype models a system intended to provide communications within a network for such participants as a composite warfare commander, antiair warfare commander, antisubmarine warfare commander, strike warfare commander, or a force coordinator. Examples of assumptions followed in designing this CAPS prototype are: retrieval of 1000 tracks within one second; entry of track-data messages into a track database within two seconds; contain four sensors, weapon systems, and communication links; update of weapon status each second; and track of, at maximum, 100 tracks per sensor per track[Ref 28: p. 58].

For the 2167A process, it is assumed the task effort will be simplistic enough for the contractor to provide a realistic cost estimate to meet the statement of work via a

fixed price (FP) contract, also known as fixed costs with deliverable type contract. In a FP contract, maximum risk is placed upon the contractor because any cost above what is negotiated is his responsibility to cover. This contract type imposes minimum management burden on the Government since what the contractor actually spends in the effort does not impact what the final payment will be. Conversely, there is a high contractor management overhead to keep track of costs and remain within budget. To compensate for this overhead, the contractor may bid a higher price, as well as include an added cushion for insurance against risk of cost overruns.

For the prototyping method, the requirements analysis and feasibility study are performed with the support of computer-aided prototyping. Because the contract will be used to provide the prototyping service, that is, running CAPS, a fee for service is desired and a cost-type contract is applied. With a cost-type contract, higher risk is placed upon the Government because it is responsible to cover any costs incurred by the contractor. However, this increased risk would be more acceptable given the computer support for the prototyping.

The total cost in dollar amount that is required to perform the tasking through each method is the measure for the cost analysis. The Air Force's Management Information Systems Technical Services (MISTS) contract, administered out of Wright-Patterson Air Force Base, Ohio was used for guidance in preparing a pseudo-task proposal and cost estimate for this evaluation. It identifies the various contractor positions used to carry out such task functions as well as the cost per hour for these positions. The Appendix contains a copy of this MISTS documentation.

B. TASK PROPOSAL AND ESTIMATE FOR THE C3I PROTOTYPE SYSTEM REQUIREMENTS UNDER THE DOD-STD-2167A PROCESS

1. Background

This task proposal is submitted as an estimate for the time, labor, and services required to complete a requirements analysis and feasibility study for an automated Command, Control, Communications, and Intelligence (C3I) system as prototyped by the Computer Science Department at the Naval Postgraduate School, Monterey, California. This proposal presents task descriptions in a very general sense, along with cost and schedule details required to define the overall project. It is assumed that this task will be a *fixed costs with deliverable* type contract with the contractor providing overhead management of the project. Table 1 provides contract line item (CLIN), work title, wage rate, and number of hours allocated for each task[Ref. 29]. The present contact for the C3I prototype was consulted along with a second individual within the Computer Science Department to estimate the effort it would require to contract out these tasks through the traditional acquisition method.

2. Scope

The scope of this task encompasses the interviewing of several personnel who will be associated with the use and development of this system along with the development of the required data-flow diagrams, data dictionaries, system specification,

TABLE I: COST ESTIMATE

CLIN AND TITLE	RATE	#HRS	TOTAL
001AA Site Manager	$57.90	360	$20844.00
003AA Sr Sys Analys	$41.01	360	$14763.60
019AA Functl Analyst	$37.84	360	$13622.40
005AA Sr Programmr	$41.65	360	$14994.00
026AA Tech Writer	$24.01	360	$8643.60
027AA Tech Typist	$13.62	360	$4903.20
			$77770.80

functional design specification, and other documents that would be required for this phase of software development under DoD-Std-2167A and any other applicable standards. Time will also be estimated for the preliminary design review, software requirements review, and any technical interchange meetings as well as the preparation by the contractor for these sessions. The development of any specific software and databases required to complete this project will also be added to the cost of this task.[Ref. 29]

3. Technical and Procedural Approach

The contractor will perform the following tasks in support of this project.

a. Overall Site Management

A site manager is required for the overall coordination and management of the project when three or more individuals are assigned, as is necessary with the number required for this task.[Ref. 29]

b. Establishment of Contractor Facility

The contractor will rent and establish work space not provided by the government to house all personnel required to complete this project. Space rental and administration fees will be taken from management overhead included in the contract for hiring a site manager.[Ref. 29]

c. Requirements Analysis and Feasibility Study

The contractor positions required for this phase of the project are the senior systems analyst, the functional analyst, and the senior programmer. A senior systems analyst is required to actually guide and perform the study. The contractor will use two senior systems analysts (CLIN 003AA) to first perform a requirements analysis for ascertaining the user's system needs, then to follow with a feasibility study of the project, constructing the appropriate data flow diagrams, a rough data dictionary, and state transition diagrams for the real-time system. The two analysts will then verify the results after the requirements analysis and feasibility study are completed by conducting the appropriate reviews specified within the contract, including DoD-Std-2167A. A functional analyst (an expert of C3I systems familiar with the concept of its system design) will assist the team in the development of questions and verification of results. In addition, a senior programmer will be required for the final portion of the project to assist in determining if the project could feasibly provide a viable product which the company is capable of producing. That is, the senior programmer will test the feasibility of the project.[Ref. 29]

d. Documentation

A documentation specialist (technical writer) in conjunction with a technical typist, will consult with the systems analyst and functional analyst to finalize the project by developing and producing the appropriate reports, briefings, and documents as required by this task. The contracting officer's technical representative will verify compliance with this agreement.[Ref. 29]

C. ESTIMATE FOR THE C3I PROTOTYPE SYSTEM REQUIREMENTS UNDER THE RAPID PROTOTYPING METHOD

1. Background

This estimate includes the time, labor, and services required to complete the requirements analysis and feasibility study for the automated C3I system using the computer-aided prototyping of the CAPS. If performed by a contractor, it is assumed that this task would be a *fee for service*, cost-type contract. Table II provides the contract line item, work title, wage rate, and number of hours allocated for each task[Ref. 29].

2. Scope

The scope of this task encompasses the interaction between the customer who will be associated with the use of such a system along with the assigned contractor personnel, who will convert the user requirements through computer-aided support into data-flow diagrams, PSDL, generation of Ada source code, and documentation as contractually required.

TABLE II: COST ESTIMATE

CLIN AND TITLE	RATE	#HRS	TOTAL
003AA Sr Sys Analys	$41.01	180	$14763.60
005AA Sr Programmr	$41.65	160	$6664.00
			$21427.60

In comparison to the previous approach, several of the personnel are unnecessary, leaving only the senior systems analyst and senior programmer as required. Because the CAPS supports creating the specification language, including timing constraints, and the graphics for data flow diagrams, it is assumed that only one senior systems analyst will be required, rather than two. The position of senior programmer is still necessary to produce the CAPS model and to make any necessary modifications to the prototyping model during the iterative process of refining user requirements. A site manager is not required as this effort will only require these two contractor positions. The functional analyst is unnecessary because the user is interacting directly with the systems analyst via the prototyping model. The positions of technical writer and typist are unnecessary because documentation of requirements will be conveyed via the specification language and graphical data flows of the prototyping model itself.[Ref. 29]

3. Technical and Procedural Approach

The contractor will perform the following tasks in support of this project.

a. Requirements Analysis and Feasibility Study

The contractor will use one senior systems analyst (CLIN 003AA) to perform a requirements analysis to ascertain the user's system needs, and will document these requirements and conduct a feasibility analysis through creation of the textual specification language (PSDL) and construction of the appropriate data flow diagrams using the CAPS. The senior programmer will have the task of creating that portion of the Ada code not automatically generated by the CAPS. The senior analyst will then verify the results after the study by conducting the appropriate reviews as required by the contract. In addition, the programmer will assist during the final portion of the project to determine if the project could feasibly provide a viable product which the company is capable of producing.[Ref. 29]

D. COMPARISON OF RESULTANTS FROM THE DOD-STD-2167A PROCESS AND COMPUTER-AIDED RAPID PROTOTYPING

The comparison of totals for effort required under each method equated to a cost reduction under the computer-aided prototyping method of $56343.20, roughly a 1:3.6 ratio or a cost savings of 27.55%. This is made with the following considerations. First, the hours allocated for senior systems analyst are decreased by half with the requirement for only one individual filling the position rather than two. Second, the senior programmer requires less effort to code the prototype because a portion of the code is generated automatically by the CAPS. For the coding done upon the C3I prototype by the Computer Science Department of the NPGS, the time involved equated

to approximately one man month of pure programmer coding effort, that is, without consideration of effort required to formulate system requirements and correction of problems with system tools[Ref. 28:p. 66]. By converting one man month to approximately 160 man hours, this is a decrease of slightly over half from the method lacking computer-aided prototyping.

E. PROJECTION OF COST SAVINGS TO A REAL-TIME SYSTEM

The initial stages of a typical program, from requirements definition through preliminary design, which are considered under the system design portion of software development, constitute roughly 13% of the total software cost[Ref. 11: p. 92-93]. Taking the results from the comparison of coding the Generic C3I Workstation, this may be projected to a mission oriented software system used in today's operations to provide a concept of what savings could be incurred when a system program office manages a software program or performs an engineering change within that operational program.

The command and control segment or CCS (described in chapter one of this thesis) is a viable candidate to project cost savings toward as it is a highly complex, mission critical computer system. Its initial contract effort under the Data Systems Modernization Contract was implemented at a cost of roughly $450 million[Ref. 30]. Of that program cost, if one can assume that the software portion of the program was the major cost driver, then 75% or $337.5 million may be considered a reasonable percentage of cost relating to the software. If one follows the concept that 13% of this results from system

design costs, then $43.875 million is the equivalent value. A cost savings of 27.55% for this program would then equate to a dollar reduction of $12.0876 million.

One could also apply this concept to an engineering change (which involves the project milestones as described in DoD-Std-2167) for a computer software configuration item (CSCI) within the CCS. An example is a change to one of the CCS's orbit software CSCIs. A typical engineering change could run $700,000[Ref. 31]. Using the estimate of 13% of the total cost for requirements definition through preliminary design, this equates to $91,000 spent for the system design portion of an engineering change of normal effort. A cost savings of 27.55% for this engineering change would then equate to a dollar reduction of $25,070.5.

VI. CONCLUSIONS

The following is a summary of the findings when comparing the traditional method of acquisition and that of computer-aided prototyping. For conducting requirements analysis and feasibility study for a Generic C3I Workstation, totals to complete the required tasks under each method equated to a cost reduction of $56343.20, using the computer-aided prototyping method. This is roughly a 1:3.6 ratio or a cost savings of 27.55%. Taking the results from this comparison, then projecting to a mission critical software system, the command and control segment (CCS), a cost savings of $12.0876 million was calculated. Finally, applying this concept to an engineering change to the CCS software showed a cost savings of $25,070.5.

This result supports the usage of computer-aided rapid prototyping to reduce acquisition costs in the early stages of a program. The assumptions which were made (reference Chapter V) along with the usage of a prototype as the model for cost comparison with projection of the results to a real-time system, prevents this thesis from providing a true dollar amount or percentage of cost a program office saves if a CAPS is used. However, the results do support the opinion that a CAPS may provide a significant enough reduction in software acquisition costs that further research and continued enhancement of the CAPS tool is justified.

LIST OF REFERENCES

1. Luqi, *Computer-Aided Prototyping of Real-Time Systems*, presented at the AFOSR/ARO/ONR Research Review on Formal Methods in Software Engineering: Concurrent and Real-Time Systems, Monterey, California, May 20-22, 1992.

2. Luqi, *Real-Time Scheduling for Requirements Specification and Testing in C3 System Acquisition*, an internal working document.

3. SS-SCF-00092B, *Air Force Satellite Control Facility System Specification for the Command and Control Segment*, April 10, 1987.

4. Department of Defense, Military Standard, DoD-STD-2167A, *Defense System Software Development*, February 29, 1988.

5. Luqi, "Software Evolution Through Rapid Prototyping", *IEEE Computer*, vol. 22, p. 13-25, May, 1989.

6. *Crosstalk The Journal of Defense Software Engineering*, published by the Software Technology Support Center, no. 39, December 1992.

7. Babich, W.A., *Software Configuration Management; Coordination for Team Productivity*, published by Addison-Wesley Publishing Company, 1986.

8. *Software Management News*, published by Software Maintenance News Inc., vol. 11, no. 1, January-February 1993.

9. Department of Defense, Military Standard, MIL-STD-480B, *Configuration Control -- Engineering Changes, Deviations, and Waivers*, July 15, 1988.

10. Department of Defense, Military Standard, MIL-STD-490A, *Specification Practices*, June 4, 1985.

11. Roetzheim, W.H., *Developing Software to Government Standards*, published by Prentice-Hall, Inc., 1991.

12. Department of Defense, Military Standard, MIL-STD-1521B, *Technical Reviews and Audits for Systems, Equipments, and Computer Software*, June 4, 1985.

13. Luqi, "A Graph Model for Software Evolution", *IEEE Trans. on Software Eng. 16*, p. 917-927, August 8, 1990.

14. *C3I Reusable Specifications, Final Technical Report*, published by International Software Systems Inc., Rome Laboratory, New York, June 1992.

15. Kieback, A., Lichter, H., Schneider-Hufshchmidt, M., and Heinz, Z., *Prototyping in Industrial Software Projects*, undated.

16. Budde, R., Kautz, K., Kuhlenkamp, K., and Heinz, Z., "What is Prototyping?", *Prototyping - an Approach to Evolutionary System Development*, Springer-Verlag, 1991.

17. Luqi, Steigerwald R., Hughes G., Naveda F., and Bergins V., "CAPS as a Requirements Engineering Tool", *Proc. 1991 Tri-Ada Conference*.

18. Anderson, S. E., *Functional Specification for a Generic C3I Workstation*, M.S. Thesis, Naval Postgraduate School, Monterey, California, September 1990.

19. Lee, Y., Luqi, and Berzins, V., *Systematic Development of Hard Real-time Software: A Comparative Study of Three Methods*, Naval Postgraduate School, Monterey, California, April, 1992.

20. Goldberg, J., *Symposium High Cost of Software*, 1973.

21. *Crosstalk The Journal of Defense Software Engineering*, published by the Software Technology Support Center, no. 36, September, 1992.

22. *Crosstalk The Journal of Defense Software Engineering*, published by the Software Technology Support Center, no. 38, November, 1992.

23. Yourdon, E., *Modern Structured Analysis*, published by YOURDON Press, 1989.

24. Goldsack, S.J. and Finkelstein, A.C.W., "Requirements Engineering for Real-time Systems", *Software Engineering Journal*, May 1991.

25. Loucopoulos, P. and Champion, R.E.M., "Concept Acquisition and Analysis for Requirements Specification, *Software Engineering Journal*, March, 1990.

26. Duvall, L. M., *RADC Software Acquisition Management and Analysis*, Syracuse University, April, 1989.

27. Hilal, D.K. and Solton, H., "To Prototype or not to Prototype? That is the Question", *Software Engineering Journal*, November, 1992.

28. Luqi, "Computer-Aided Prototyping for a Command and Control System Using CAPS", *IEEE*, p. 56-67, January, 1992.

29. Interview between D. Gaitros, Major, USAF, Computer Science Department, Naval Postgraduate School, Monterey, California, and the author, 6 May 93.

30. Interview between E. Garcia, Civilian, USAF, Contracts Administrator, Onizuka Air Force Base, Sunnyvale, California, and the author, 14 May 93.

31. Interview between P. Price, Contractor, Applied Technology Associates, Configuration Management, Mountain View, California, and the author, 14 May 93.

BIBLIOGRAPHY

1. "Back to the Future Through Reengineering", *First Software Reengineering Workshop*, Red Lion Inn, Santa Barbara California, 25 September 1992.

2. *Conferences, Seminars, and Services*, published by Software Quality Engineering, catalog, vol. 1,1993.

3. Department of Defense, *Chairman of the JCS Report on the Roles, Missions, and Functions of the Armed Forces of the United States*, February 1993.

4. *Single Source*, published by Software Quality Engineering, 1993.

5. "Software Engineering Technical Committee Newsletter", published by the IEEE Computer Society/TCSE, vol. 11, no. 2, January 1993.

6. "Software Engineering Technical Committee Newsletter", published by the IEEE Computer Society/TCSE, vol. 11, no. 3, January 1993.

7. Tran, N.T. and Mumm, R.H., *Environment/Tool Integrator for Software Development, Technical Report 1548*, published by Naval Command, Control, and Ocean Surveillance Center, San Diego, California, version 1.1, August 1992.

8. Kemerer, C. F., "An Empirical Validation of Software Cost Estimation Models", *Communications of the ACM*, vol. 30, no. 5, May, 1987.

APPENDIX

Task Proposal and Estimate for the C^3 I Prototype System Requirements Using the Air Force Management Information Systems Technical Services (MISTS) Contract

1.0 BACKGROUND

This task proposal is submitted as an estimate for the time, labor and services required to complete a requirements analysis and feasibility study for an automated Communications, Command, Control and Intelligence system as prototyped by the Computer Science Department at the Naval Postgraduate School, Monterey, California. This proposal presents task descriptions in a very general sense, along with cost and schedule details required to define the overall project. It is assumed that this task will be a fixed costs with deliverable type contract with the contractor providing overhead management of the project.

2.0 SCOPE

The scope of this task encompasses the interviewing of several personnel who would be associated with the use and development of such a system along with the development of the required data-flow diagrams, data dictionaries, functional design specifications, preliminary design specifications and other documents that would be required for this phase of software development under DoD 2167A and other applicable regulations. Time will also be estimated for functional and preliminary design reviews and the preperation of such reviews by the contractor. The development of any specific software and databases required to complete this project will also be added to the cost of this task.

3.0 TECHNICAL AND PROCEDURAL APPROACH

The contractor will perform the following tasks in support of this project.

3.1 Establishment of Contractors Facility.

The contractor shall rent and establish work space not provided by the government to house all personnel required to complete this project. Space rental and adminstration fees will be taken from management overhead included in the contract for hiring a site manager.

3.2 Feasibility Study

The contractor will use two senior systems analyst (clin 03A) to conduct a feasibility study of the project, construct the appropriate data flow diagrams, construct a rough data dictionary and state transition diagrams for the real time systems. The two analyst will then verify the results after the study by conducting the appropriate reviews as required by the contract and DoD 2167A. A functional specialist familiar with C^3I systems will assist the team in the development of questions and verification of the results. In addition, a senior programmer will be required for the last portion of the project to assist in determining if the feasibility of the project is a viable product in which the company is capable of producing.

3.3 Documentation

A documentation specialist (technical writer), the sytsems analyst, the functional analyst and the technical typist will finalize the project by developing and producing the appropriate reports, briefings and documents as required by this task. The contracting officers technical representative will verify compliance with this agreement.

COST ESTIMATE

CLIN	DESCR01	RATE	# HOURS	TOTAL
01AA	SITE MANAAGER	57.90	360	
03AA	SEN SYSTEM ANAL	41.01	360	
03AA	SEN SYSTEM ANAL	41.01	360	
019AA	FUNCTIONAL ANAL	37.84	360	
05AA	SENIOR PROGRAM	41.65	240	
026AA	TECH WRITER	24.01	240	
027AA	TECH TYPIST	13.62	240	
			TOTAL	

MANAGEMENT INFORMATION SYSTEMS TECHNICAL SERVICES (MISTS)

TASK REQUEST

Task Name: HQ AFSC and HQ AFMC Communications-Computer Systems (C-CS) Transition Planning and Management

Task Type: Work Order

1.0 Background:

To support forming Air Force Material Command, HQ AFSC/SC is involved with HQ AFMC/SC in C-CS transition planning, and transferring assets and functions to HQ AFMC.

The first major focus is to support developing the detailed transition plans to ensure that the C-CS infrastructure is efficiently transitioned to Wright-Patterson AFB, and in operation by July 1992. An overall program plan (P-Plan), for which SC activities are annexes, governs this transition. This planning also involves phasing out C-CS support systems within the AFSC HQ complex at Andrews AFB MD as well as working with the new building occupant to transition facilities and equipment which will remain in place after July 1992.

Under a previous task order, TRW personnel developed a transition database which includes information on the status of transition-related requirements. This database assists The AFMC Provisional Headquarters (AFMC(P)/SC) in monitoring transition activities. Contractor support of transition planning and maintenance of this database will continue through July 92.

During the last six months HQ AFSC government and contractor personnel have also supported planning for the HQ AFMC Office Automation Network. This support activity includes developing program plans, support plans, training plans, implementation plans, acquisition plans and budgets. This work is in support of the Office Automation Program Office established at HQ AFLC. The work within HQ AFSC is expected to continue in order to ensure that HQ AFSC requirements are reflected in the OA plans and technical solutions.

The second major focus is to oversee executing transition plans to assure that assets and functions related to the mini-computers in user office spaces are being efficiently and completely transitioned. Oversight may encompass activities at both ends: HQ AFSC and HQ AFMC.

2.0 Scope of Work. The scope of this task encompasses the activities described above. It includes providing direct technical input to the transition, and OA planning processes. It includes supporting the production, editing and maintenance of functional C-CS transition plans, and supporting continuing OA planning and requirements analysis activities as the HQ AFMC OA system expands from an initial base of 200 to 2000 users. The scope also encompasses those activities that are necessary to completely oversee transition activity that is related to functional mini-computers.

3.0 Functional/Technical Requirements.

3.1 Transition Planning Support. The contractor shall make modifications to the transition requirements database at the request of AFMC(P)/SC. The contractor shall assist in preparing detailed functional C-CS transition plans and schedules including annexes to the P-Plan. The contractor shall assist in tracking transition activities and schedules and prepare status briefings and reports supporting government reviews. The contractor shall assist in planning the phase out and transition of HQ AFSC C-CS support equipment and facilities to the new building occupant. Planning materials, reports, and schedules shall be generated and maintained as required to manage and track transition activities.

3.2 Office Automation Planning. The contractor shall develop and maintain the OA C-CS Program Plan, Support Plan, and other plans and specifications as directed by the COTR. The contractor shall assist in preparing budgets and supporting documentation required for POM inputs. The contractor shall review programmatic documentation for technical sufficiency and to ensure that AFSC functional requirements are adequately supported. The contractor shall serve as a liaison between the HQ AFMC OA Program Office and HQ AFSC/SC. The contractor shall assist in analyzing and documenting interfaces between the OA system and functional stove pipe systems. Based on the analysis, alterative user interfaces supporting the one button access concept shall be developed and documented as technical reports.

3.3 Mini-Computer Transition Oversight. The contractor shall supervise personnel who are executing transition plans and coordinate the move activities for AT&T 3B2s together with related office communications equipment, located in HQ AFSC SG, XT, LG, DP, MO, and SC. The SC 3B2s include those currently located in the Computer Operations and Systems Management Engineering Center (COSMEC). Other mini-computers may include the Wang VS-100 in HQ AFSC/DE, and Sperry 2200-200/400 that are located in HQ AFSC/FM and the COSMEC.

4.0 Period of performance.

4.1 Start Date: 1 October 1991

4.2 Completion Date: 30 June 1992 for tasks 3.1 and 3.2; 30 September 1992 for task 3.3.

5.0 Deliverables and Standards. With the exception of technical reports and status reports, no specific deliverables shall be required under this task. The contractor shall assist the Air Force in producing draft and final planning documents, analyses, reports briefings and schedules. Document production will take place as an iterative process through exchanging drafts. Final document production will be the responsibility of the Air Force.

Axxx	Technical Report	As Required
Axxx	Monthly Status Report	Monthly

6.0 Acceptance Criteria. Deliverable format and content outlines will be approved by the COTR prior to beginning work on the deliverable. Deliverable content and format must comply with COTR approved direction.

7.0 Place of Primary Performance. HQ AFSC (on site).

8.0 Hours of Work. Contractor's normal duty hours.

9.0 Travel Requirements. Periodic trips to Wright-Patterson AFB OH will be required for planning and coordination meetings, and oversight. For planning purposes, assume 10 trips for 2 people for 5 days each to Wright-Patterson AFB, OH.

10.0 Points of Contact.

10.1 COTR for this task:	Mr Mike Tervo, HQ AFSC/SCT
10.2 User Project Officer:	Maj Dave Gaitros, HQ AFSC/SCT Mr Mike Tervo, HQ AFSC/SCT 1Lt Randal Taylor, HQ AFSC/SCT Maj Al Weimer, LMSC/SXNS Mr Mike Riley, LMSC/SXNS
10.3 Other Points of Contact:	Maj H McCoy, HQ AFMC(P)/SC

11.0 Special Required Personnel Qualifications: Contractor Personnel shall be skilled C-CS Planners and possess a working-level knowledge of program management, and the documentation required with which to oversee a project. Furthermore, the contractor personnel should be familiar with office automation architectures, processes, and procedures. A knowledge of current AFSC business practices, C-CS planning strategies, and C-CS architectures is desirable. Contractor personnel shall also be skilled in management and possess people handling skills to ensure maximum harmony is achieved with all parties who are moving the mini-computers.

12.0 Security Clearance Requirement. None.

13.0 Communications, Hardware, and Software Environment. Access to the HQ AFSC EIS, or the HQ AFLC LOGDIS system shall be made available to project personnel for electronic communications with the headquarters and field. At least two Desktop-III, or equivalent, PC running the Microsoft office software suite shall be provided. Remaining personnel shall be supplied Z-248, or equivalent, PCs.

14.0 Special Government Provided Facilities, Services and Supplies. The Government shall provide office space, office supplies and telephone service for at least two personnel.

15.0 Attachments. None.

BASIC PERIOD OF PERFORMANCE
01 OCT 91 through 30 SEP 92
AREA 1 Locations: Andrews AFB

Subcontractor: Booz, Allen & Hamilton

CLIN	DESCRIPTION	EST QTY	UNIT	UNIT PRICE
0001AA	Software Development Mgr	1,920	Hr.	$0.00
0002AA	Operations & Support Mgr	1,920	Hr.	0.00
0003AA	Senior Systems Analyst	9,600	Hr.	0.00
0004AA	Systems Analyst	5,760	Hr.	0.00
0005AA	Senior Programmer	0	Hr.	38.50
0006AA	Programmer	17,280	Hr.	0.00
0007AA	Training Supervisor	1,920	Hr.	0.00
0008AA	Trainer	11,520	Hr.	0.00
0009AA	Senior Communications Eng	1,920	Hr.	0.00
0010AA	Communications Engineer	960	Hr.	33.61
0011AA	Communication Technician	8,640	Hr.	0.00
0012AA	Comp. Resources Admin.	5,760	Hr.	0.00
0013AA	Comp Ops Supervisor	0	Hr.	0.00
0014AA	Senior Computer Operator	5,760	Hr.	0.00
0015AA	Computer Operator	0	Hr.	0.00
0016AA	Help Desk Coordinator	7,680	Hr.	0.00
0017AA	Field Service Technician	9,600	Hr.	0.00
0018AA	Field Service Rep	1,920	Hr.	0.00
0019AA	Systems Eng. Specialist	3,648	Hr.	0.00
0020AA	Data Base Mgmt. Specialist	3,648	Hr.	0.00
0021AA	Office Automation Spec.	768	Hr.	37.01
0022AA	Config. Mgmt Specialist	3,840	Hr.	0.00
0023AA	Comp. Graphics Specialist	768	Hr.	0.00
0024AA	Systems Software Specialist	1,920	Hr.	0.00
0025AA	ADP Hardware Specialist	576	Hr.	0.00
0026AA	Technical Writer	5,568	Hr.	0.00
0027AA	Technical Typist	1,728	Hr.	0.00
0028AA	Data Entry Clerk	1,920	Hr.	0.00
0029AA	Data Control Clerk	192	Hr.	0.00
0030AA	Government Office Space	N/A		N/A
0031AA	Contractor Office Space	N/A		N/A
0032AA	DATA	N/A		NSP
		116,736		

N/A = Not Applicable
NSP = Not Separately Priced

NOTE: Basic year performance period will be adjusted based on date of Award. Frequency of CLIN 0032 - Reports, shall be in in accordance with CDRL A022 and Section F.5.

CSC

INFORMATION SET FORTH ON THIS PAGE IS SUBJECT TO THE RESTRICTION PRINTED ON THE TITLE PAGE OF THIS PROPOSAL

BASIC PERIOD OF PERFORMANCE
01 OCT 91 through 30 SEP 92
AREA 1 Locations: Andrews AFB

Subcontractor: Century Technologies, Inc. (CENTECH)

CLIN	DESCRIPTION	EST QTY	UNIT	UNIT PRICE
0001AA	Software Development Mgr	1,920	Hr.	$0.00
0002AA	Operations & Support Mgr	1,920	Hr.	0.00
0003AA	Senior Systems Analyst	9,600	Hr.	0.00
0004AA	Systems Analyst	5,760	Hr.	0.00
0005AA	Senior Programmer	0	Hr.	32.59
0006AA	Programmer	17,280	Hr.	0.00
0007AA	Training Supervisor	1,920	Hr.	25.91
0008AA	Trainer	11,520	Hr.	23.88
0009AA	Senior Communications Eng	1,920	Hr.	0.00
0010AA	Communications Engineer	960	Hr.	32.59
0011AA	Communication Technician	8,640	Hr.	0.00
0012AA	Comp. Resources Admin.	5,760	Hr.	23.88
0013AA	Comp Ops Supervisor	0	Hr.	30.42
0014AA	Senior Computer Operator	5,760	Hr.	0.00
0015AA	Computer Operator	0	Hr.	0.00
0016AA	Help Desk Coordinator	7,680	Hr.	20.35
0017AA	Field Service Technician	9,600	Hr.	0.00
0018AA	Field Service Rep	1,920	Hr.	0.00
0019AA	Systems Eng. Specialist	3,648	Hr.	0.00
0020AA	Data Base Mgmt. Specialist	3,648	Hr.	34.69
0021AA	Office Automation Spec.	768	Hr.	37.43
0022AA	Config. Mgmt Specialist	3,840	Hr.	25.91
0023AA	Comp. Graphics Specialist	768	Hr.	25.91
0024AA	Systems Software Specialist	1,920	Hr.	34.69
0025AA	ADP Hardware Specialist	576	Hr.	30.42
0026AA	Technical Writer	5,568	Hr.	0.00
0027AA	Technical Typist	1,728	Hr.	0.00
0028AA	Data Entry Clerk	1,920	Hr.	0.00
0029AA	Data Control Clerk	192	Hr.	14.02
0030AA	Government Office Space	N/A		N/A
0031AA	Contractor Office Space	N/A		N/A
0032AA	DATA	N/A		NSP
		116,736		

N/A - Not Applicable
NSP - Not Separately Priced

NOTE: Basic year performance period will be adjusted based on date of Award. Frequency of CLIN 0032 - Reports, shall be in in accordance with CDRL A022 and Section F.5.

FOR DISCLOSURE OF PROPOSAL DATA SEE RESTRICTION PROVISION ON THE TITLE PAGE OF THIS PROPOSAL

BASIC PERIOD OF PERFORMANCE
01 OCT 91 through 30 SEP 92
AREA 1 Locations: Andrews AFB

Subcontractor: Digital Equipment Corporation

CLIN	DESCRIPTION	EST QTY	UNIT	UNIT PRICE
0001AA	Software Development Mgr	1,920	Hr.	$0.00
0002AA	Operations & Support Mgr	1,920	Hr.	0.00
0003AA	Senior Systems Analyst	9,600	Hr.	0.00
0004AA	Systems Analyst	5,760	Hr.	0.00
0005AA	Senior Programmer	0	Hr.	0.00
0006AA	Programmer	17,280	Hr.	0.00
0007AA	Training Supervisor	1,920	Hr.	0.00
0008AA	Trainer	11,520	Hr.	0.00
0009AA	Senior Communications Eng	1,920	Hr.	0.00
0010AA	Communications Engineer	960	Hr.	0.00
0011AA	Communication Technician	8,640	Hr.	0.00
0012AA	Comp. Resources Admin.	5,760	Hr.	0.00
0013AA	Comp Ops Supervisor	0	Hr.	0.00
0014AA	Senior Computer Operator	5,760	Hr.	0.00
0015AA	Computer Operator	0	Hr.	0.00
0016AA	Help Desk Coordinator	7,680	Hr.	0.00
0017AA	Field Service Technician	9,600	Hr.	0.00
0018AA	Field Service Rep	1,920	Hr.	0.00
0019AA	Systems Eng. Specialist	3,648	Hr.	0.00
0020AA	Data Base Mgmt. Specialist	3,648	Hr.	0.00
0021AA	Office Automation Spec.	768	Hr.	67.17
0022AA	Config. Mgmt Specialist	3,840	Hr.	0.00
0023AA	Comp. Graphics Specialist	768	Hr.	0.00
0024AA	Systems Software Specialist	1,920	Hr.	0.00
0025AA	ADP Hardware Specialist	576	Hr.	0.00
0026AA	Technical Writer	5,568	Hr.	0.00
0027AA	Technical Typist	1,728	Hr.	0.00
0028AA	Data Entry Clerk	1,920	Hr.	0.00
0029AA	Data Control Clerk	192	Hr.	0.00
0030AA	Government Office Space	N/A		N/A
0031AA	Contractor Office Space	N/A		N/A
0032AA	DATA	N/A		NSP
		116,736		

N/A = Not Applicable
NSP = Not Separately Priced

NOTE: Basic year performance period will be adjusted based on date of Award. Frequency of CLIN 0032 - Reports, shall be in in accordance with CDRL A022 and Section F.5.

THIS SHEET OF PROPOSAL DATA IS SUBJECT TO THE RESTRICTION CONTAINED ON THE TITLE PAGE OF THIS PROPOSAL.

BASIC PERIOD OF PERFORMANCE
01 OCT 91 through 30 SEP 92
AREA 1 Locations: Andrews AFB

Subcontractor: Sumaria Systems, Inc.

CLIN	DESCRIPTION	EST QTY	UNIT	UNIT PRICE
0001AA	Software Development Mgr	1,920	Hr.	
0002AA	Operations & Support Mgr	1,920	Hr.	
0003AA	Senior Systems Analyst	9,600	Hr.	
0004AA	Systems Analyst	5,760	Hr.	
0005AA	Senior Programmer		Hr.	
0006AA	Programmer	17,280	Hr.	
0007AA	Training Supervisor	1,920	Hr.	
0008AA	Trainer	11,520	Hr.	
0009AA	Senior Communications Eng	1,920	Hr.	
0010AA	Communications Engineer	960	Hr.	
0011AA	Communication Technician	8,640	Hr.	
0012AA	Comp. Resources Admin.	5,760	Hr.	
0013AA	Comp Ops Supervisor		Hr.	
0014AA	Senior Computer Operator	5,760	Hr.	
0015AA	Computer Operator		Hr.	
0016AA	Help Desk Coordinator	7,680	Hr.	22.91
0017AA	Field Service Technician	9,600	Hr.	
0018AA	Field Service Rep	1,920	Hr.	
0019AA	Systems Eng. Specialist	3,648	Hr.	
0020AA	Data Base Mgmt. Specialist	3,648	Hr.	
0021AA	Office Automation Spec.	768	Hr.	48.05
0022AA	Config. Mgmt Specialist	3,840	Hr.	
0023AA	Comp. Graphics Specialist	768	Hr.	
0024AA	Systems Software Specialist	1,920	Hr.	
0025AA	ADP Hardware Specialist	576	Hr.	42.74
0026AA	Technical Writer	5,568	Hr.	
0027AA	Technical Typist	1,728	Hr.	
0028AA	Data Entry Clerk	1,920	Hr.	
0029AA	Data Control Clerk	192	Hr.	
0030AA	Government Office Space	N/A		N/A
0031AA	Contractor Office Space	N/A		N/A
0032AA	DATA	N/A		NSP
		116,736		

N/A = Not Applicable
NSP = Not Separately Priced

NOTE: Basic year performance period will be adjusted based on date of Award. Frequency of CLIN 0032 - Reports, shall be in in accordance with CDRL A022 and Section F.5.

DISCLOSURE OF PROPOSAL DATA IS SUBJECT TO THE RESTRICTION PRINTED ON THE TITLE PAGE OF THIS PROPOSAL.

Tplanprp.WP

TASK PROPOSAL

MANAGEMENT INFORMATION SYSTEMS TECHNICAL SERVICES (MISTS)

HQ AFSC and HQ AFMC Communications-Computer Systems Transition Planning and Management

Contract Number F49642

September 18, 1991

Prepared for:
HQ AF Systems Command
Deputy Chief of Staff Communications-Computer Systems
Andrews AFB MD 20334

TABLE OF CONTENTS

1. BACKGROUND

This task proposal is submitted to implement the AFSC/SC task request for transition planning and management support. This proposal presents task descriptions and cost and schedule details to define tasks required to support SC in planing and managing the varied activities required to transition systems and C-CS support from to AFSC to HQ AFMC.

Overall guidance for the transition is supplied by PAD 91-6 which directed the integration of AFSC and AFLC to form AFMC. Based on the PAD, the HQ AFLC/AFSC PPlan 91-01, dated June 1991 was generated. The PPlan provides further details of actions and schedules necessary to form AFMC by 1 Jul 92.

To specifically address C-CS planning issues, in March 1991 a joint AFSC/AFLC Transition Team was formed to meet with individual functional organizations to determine which C-CSs were required to support AFMC on 1 Jul 92. Requirements derived from a series of meetings were documented in a final report and a transition database. HQ AFMC(P)/SC has used the team's finding to develop a C4 Beddown Plan which is currently being reviewed within both commands. In addition to the beddown plan, more detailed transition plans for specific systems are being jointly developed by SC and functional users. Each plan will be approved by Memoranda of Understanding between the user and all supporting organizations. Each plan will include detailed activities, schedules and responsibilities. After MOA approval, government personnel, supported by contractors if required, will manage all activities necessary to complete transition to AFMC.

CSC personal who will be supporting this task are extremely knowledgeable of the full scope of transition requirements. Working under a previous support contract Mr Carpenter served as a member of the transition team. Mr Hamrick developed the transition database and generated detailed transition plans, schedules and MOAs for transitioning minicomputer systems.

In parallel with the transition planning, an effort to implement a significantly enhanced office automation capability for HQ AFMC was initiated by the AFMC Provisional HQ Commander. Mr Carpenter was jointly designated by HQ AFSC/SC and HQ AFLC/SC as the team leader for this project. The initial task focused on installing a client-server network and 70 personnel computers within the Provisional Headquarters. Expansion of the initial capability to support the command section and all 2-letter front offices in now underway. It is due to be completed in October. Phase 2 of the effort will expand the system to support approximately 2000 users throughout the headquarters.

An Office Automation Program Office has been established within

AFLC to oversee the design and installation of the network as well as supply operations, maintenance, training, network management and customer support for the office automation initiative. Through July of 92 there will be a continuing need to ensure AFSC personnel are involved in the guiding the definition and implementation of the network. This includes reviewing specifications and other programmatic documents and participating in configuration control board meetings and design reviews.

2. SCOPE

The scope of this task encompasses the transition planning and management and office automation program support described above. It includes providing direct technical input to transition plans and office automation network specifications and other documentation. It also incudes planning and management support as required to ensure transition activities take place in accordance with approved schedules.

3.0 TECHNICAL APPROACH

CSC will perform the subtasks defined below. In addition to these subtasks, there are program management activities which will take place during the life of the task to ensure technical and schedule performance requirements are met.

3.1 Transition Planning Support

CSC will assist the government in developing and refining detailed transition plans including the C4 Beddown Plan and supporting plans. CSC personnel will review and update information developed by the C-CS transition team to ensure it reflects current move schedules and evolving C-CS requirements. When necessary they will meet with functional representatives to clarify and document requirements. At the request of AFMC(P) SC the transition database will be updated. Transition requirements, schedules, and status reports will be generated using the database and other sources of information. Status briefings will be generated to support the Air Forces transition management activities.

3.2 Office Automation Planning

CSC personnel will serve as a liaison between the Office Automation Program Office and HQ AFSC/SCT. This activity will include attending and reporting on formal program reviews, Configuration Control Board meetings, and design reviews. CSC personnel provide reports to SCT covering the results of each meeting and any issues that require resolution. CSC personnel will review design documentation to ensure that AFSC's functional and performance requirements are included. CSC will assist SCT in reviewing plans and specifications for the Executive Information Systems module and the automated conference room capabilities which will be integrated into the

OA Network. Programmatic documents including the program plan, logistics support plan, configuration management plan, security plan and others will be reviewed and comments provided to SCT. At the direction of the COTR, CSC personnel will assist in generating office automation plans, supporting documents, and budgets.

4.0 Personnel

Mr Doug Carpenter (Office Automation Specialist) will serve as the task leader. Mr Carpenter served on the Transition Planning Team and was responsible for developing the final report which documented HQ-wide transition requirements. He also lead the team that developed the HQ Office Automation Implementation Plan, and served as the Office Automation Task Leader until the Program Office was in place. Mr Carpenter will serve as the primary interface with the Office Automation Program Office, he will also support continued transition planning and execution.

Mr Thomas Hamrick (Senior Systems Analyst) will also support this task. For the last two years Mr Hamrick has supported several HQ AFSC SC engineering and planning efforts. For the last six months he has specifically supported transition planning activities. He reviewed all functional requirements information and implemented a transition database which is available for use by the AFMC(P)/SC as well as AFSC planners. More recently he has assisted SCT personnel in defining detailed 3B2 transition activities, developing schedules, and drafting Memoranda of Understanding. Under this task, Mr Hamrick will continue to develop and refine transition plans as well as manage the activities necessary to successfully move systems and return them to operational status.

5. MILESTONES, DELIVERABLES, AND SCHEDULE

5.1 Deliverables

The contract deliverables for this task are as follows:

Cxxx	Technical Report	As required
Cxxx	Monthly Status Report	Monthly*

*Report will be provided by E-Mail on a schedule to be specified by the COTR.

5.2 Schedule

3.1 1 Oct 91 through 30 Jun 92
3.2 1 Oct 91 through 30 Jun 92
3.3 1 Oct 91 through 30 Sep 92

6. WORK BREAKDOWN SCHEDULE/MANPOWER LOADING

The following figure supplies the WBS and allocated labor hours and costs.

Task	Title	OA Spec	SSA	Total
A1.1	Transition Planning Support	500	680	
A1.2	Office Automation Program Support	700	200	
A1.3	Mini-Computer Transition Oversight	200	1200	
	TOTAL HOURS	1400	2080	
	RATE	55.13	40.01	
	LABOR PRICE	77,182	83,220	
	TOTAL LABOR			$160,402
TRAVEL				$21,000
	10 Trips for 2 people for 5 days to WPAFB			
TOTAL PRICE				$181,402

7. RESOURCES

7.1 Government Furnished Equipment/Information

The following equipment, information, data, and services -- jointly referred to as Government Furnished Equipment (GFE) will be provided to the CSC under this task:

(a) Standard office facilities for two professionals. These facilities will include work space and sufficient telephone equipment. It will also include standard office supplies. Document reproduction services will be supplied as required.

(b) Access to the VAX Cluster for two electronic mail accounts.

(c) Access to C-CS transition and Office Automation Program related information including regulations, plans, requirements documents, technical descriptions and schedules.

(d) Access to information on DCS transition requirements and AFMC required C-CS capabilities collected by SC personnel.

(e) Access to two DTIII PCs with the standard suite of office automation software. At least one PC shall host Microsoft Project.

7.2 Other Direct Costs

None

8.0 ASSUMPTIONS AND CONSTRAINTS

None

9.0 ACCEPTANCE

Acceptance will be based on the COTR approval of hours worked. Document acceptance shall be based on compliance with COT approved scope and format.

FY92 CLIN	1440 SKILL	CSC RATE	1440 HRS OCT-JUNE	BAH RATE	1440 HRS OCT-JUNE	CENT RATE	1440 HRS OCT-JUNE	DEC RATE	1440 HRS OCT-JUNE	SUM RATE	1440 HRS OCT-JUNE
01A	SDM	57.90	83,376.00								
02A	OSM	51.84	74,649.60								
03A	SSA	41.01	59,054.40								
04A	SA	31.85	45,864.00								
05A	SP	41.65	59,976.00	38.50	55,440.00	32.59	46,929.60				
06A	P	26.77	38,548.80								
07A	TRN SPV	37.28	53,683.20			25.91	37,310.40				
08A	TRN	24.01	34,574.40			23.88	34,387.20				
09A	SCE	37.74	54,345.60								
10A	CE	33.39	48,081.60	33.61	48,398.40	32.59	46,929.60				
11A	CT	18.53	26,683.20								
12A	CRA	24.01	34,574.40			23.88	34,387.20				
13A	CO SPV	35.08	50,515.20			30.42	43,804.80				
14A	SCO	18.45	26,568.00								
15A	CO	18.02	25,948.80								
16A	HDC	21.92	31,564.80			20.35	29,304.00			22.91	32,990.40
17A	FST	23.01	33,134.40								
18A	FSR	14.93	21,499.20								
19A	SES	37.84	54,489.60								
20A	DBMS	38.38	55,267.20			34.69	49,953.60				
21A	OAS	55.13	79,387.20	37.01	53,294.40	37.43	53,899.20	67.17	96,724.80	48.05	69,192.00
22A	CMS	41.16	59,270.40			25.91	37,310.40				
23A	CGS	39.50	56,880.00			25.91	37,310.40				
24A	SSS	37.28	53,683.20			34.69	49,953.60				
25A	AHS	43.39	62,481.60			30.42	43,804.80			42.74	61,545.60
26A	TW	24.01	34,574.40								
27A	TT	13.62	19,612.80								
28A	DE	13.23	19,051.20								
29A	DC	16.49	23,745.60			14.02	20,188.80				

BASIC PERIOD OF PERFORMANCE
01 OCT 91 through 30 SEP 92
AREA 1 Locations: Andrews AFB

Contractor: Computer Sciences Corporation

CLIN	DESCRIPTION	EST QTY	UNIT	UNIT PRICE
0001AA	Software Development Mgr	1,920	Hr.	$57.90
0002AA	Operations & Support Mgr	1,920	Hr.	51.84
0003AA	Senior Systems Analyst	9,600	Hr.	41.01
0004AA	Systems Analyst	5,760	Hr.	31.85
0005AA	Senior Programmer	0	Hr.	41.65
0006AA	Programmer	17,280	Hr.	26.77
0007AA	Training Supervisor	1,920	Hr.	37.28
0008AA	Trainer	11,520	Hr.	24.01
0009AA	Senior Communications Eng	1,920	Hr.	37.74
0010AA	Communications Engineer	960	Hr.	33.39
0011AA	Communication Technician	8,640	Hr.	18.53
0012AA	Comp. Resources Admin.	5,760	Hr.	24.01
0013AA	Comp Ops Supervisor	0	Hr.	35.08
0014AA	Senior Computer Operator	5,760	Hr.	18.45
0015AA	Computer Operator	0	Hr.	18.02
0016AA	Help Desk Coordinator	7,680	Hr.	21.92
0017AA	Field Service Technician	9,600	Hr.	23.01
0018AA	Field Service Rep	1,920	Hr.	14.93
0019AA	Systems Eng. Specialist	3,648	Hr.	37.84
0020AA	Data Base Mgmt. Specialist	3,648	Hr.	38.38
0021AA	Office Automation Spec.	768	Hr.	55.13
0022AA	Config. Mgmt Specialist	3,840	Hr.	41.16
0023AA	Comp. Graphics Specialist	768	Hr.	39.50
0024AA	Systems Software Specialist	1,920	Hr.	37.28
0025AA	ADP Hardware Specialist	576	Hr.	43.39
0026AA	Technical Writer	5,568	Hr.	24.01
0027AA	Technical Typist	1,728	Hr.	13.62
0028AA	Data Entry Clerk	1,920	Hr.	13.23
0029AA	Data Control Clerk	192	Hr.	16.49
0030AA	Government Office Space	N/A		N/A
0031AA	Contractor Office Space	N/A		N/A
0032AA	DATA	N/A		NSP
		116,736		

N/A - Not Applicable
NSP - Not Separately Priced

NOTE: Basic year performance period will be adjusted based on date of Award. Frequency of CLIN 0032 - Reports, shall be in accordance with CDRL A022 and Section F.5.

CSC

SEE DISCLOSURE OF PROPOSAL DATA IS SUBJECT TO THE RESTRICTION PRINTED ON THE TITLE PAGE OF THIS PROPOSAL

TERVO

DEPARTMENT OF THE AIR FORCE
HEADQUARTERS AIR FORCE SYSTEMS COMMAND
ANDREWS AIR FORCE BASE DC 20334-5000

REPLY TO ATTN OF: SCK

9 September 1991

SUBJECT: F49650-91-D0011, Management Information Systems Technical Service Contract (MISTS)

TO: SCT

1. Under the MISTS contract, Computer Sciences Corporation (CSC) and their team of four (4) subcontractors are totally responsible for providing a wide range of technical services at all AFSC locations.

2. The following procedure will be utilized to evaluate any CSC request to modify the MISTS contract to provide the services of additional subcontractors:

a. In accordance with the MISTS Task Request/Delivery Order Operating Procedures, Exhibit 1, a Task Request, a Government Task Estimate Form, and a funding document (attached) will be submitted by the user activity to the contracting office responsible for servicing the respective AFSC activity.

b. After review to assure that the documentation is complete, the servicing contracting office will issue the Task Request to CSC.

c. If CSC believes the scope of work defined in the Task Request can be accomplished within the existing contractual resources, CSC will submit a Task Proposal to the servicing contracting office with a copy to the user activity. Negotiation and issuance of a Delivery Order will proceed in accordance with standard MISTS operating procedures.

d. If CSC believes the scope of work defined in the Task Request is beyond the scope of the MISTS contract, the following actions will take place:

(1) CSC will submit a letter to HQ AFSC/SCK with a copy to the servicing contracting office supporting their contention that the work defined in the Task Request is beyond the scope of the MISTS contract.

(2) Further processing of the Task Request will be suspended pending resolution of the contract scope issue. HQ AFSC/SCK will notify the servicing contracting office when the issue has been resolved.

e. If CSC believes the scope of work defined in the Task Request is within the scope of the MISTS contract, but cannot be satisfactorily performed with existing contractual resources, the following actions will take place:

(1) CSC will submit a letter to HQ AFSC/SCK with a copy to the servicing contracting office documenting their inability to satisfactorily perform the work defined in the Task Request with existing contractual resources. The original copy of the letter will be accompanied by a proposed contract modification to add subcontractor(s) resources which will permit satisfactory performance.

(2) Further processing of the Task Request will be suspended pending evaluation/negotiation of the proposed contract modification. HQ AFSC/SCK will notify the servicing contracting office when the issue has been resolved.

3. Inquiries concerning this subject should be directed to me at DSN 858-5599.

JAMES W. AINSLIE
Government Program Manager, MISTS
DCS/Communications-Computer Systems

1 Atch
Exhibit 1 w/3 Atch

Exhibit 1

MANAGEMENT INFORMATION SYSTEMS TECHNICAL SERVICE CONTRACT

WITH

COMPUTER SCIENCES CORPORATION - CONTRACT # F49650-91-D0011

TASK REQUEST / DELIVERY ORDER OPERATING PROCEDURES

1.0 Scope of Contract

The Management Information Systems Technical Service (MISTS) contract provides for the acquisition of a wide range of management information system ADP and communications support services to include computer operations, planning and analysis, software development, documentation, user support, configuration management, software maintenance, training, communications systems design and installation, systems integration, etc. and equipment maintenance of end user devices such as the VT-100, VT-240, Z-248 and associated modems, peripherals, etc. The contract is a mandatory source of supply for the management information system requirements of AFSC at Andrews AFB and Bolling AFB, and may be used on an optional basis by any other AFSC activity.

2.0 Contract Data

2.1 Contract Type. Fixed rate time and materials requirements contract.

2.2 Contract Term. The base period of the contract is from 1 October 1991 through 30 September 1992, with four one-year priced renewal options.

2.3 Maximum Delivery Order Limitation. $4 million.

3.0 Order Processing

Contract services are ordered by the issuance of negotiated delivery orders which specify defined scopes of work, schedules for completion, technical requirements, performance standards, criteria for deliverable products, and total price. Delivery order requirements may range from a need for a small number of contractor personnel to be temporarily located at an AFSC location in order to perform a specific task, to the requirement for a large number of contractor personnel to be permanently located at HQ AFSC or at any other AFSC location where there is a substantial continuous workload.

4.0 Government Task Request

The initial step in the ordering process is the creation of a Task Request. A Task Request format is enclosed as Attachment 1 and may be submitted as a project task or a work order task.

4.1 Project Task. A project task is a task for which there are reasonably

well defined requirements. These will be issued on a firm-fixed-price basis. Software development tasks (detail systems design, programming, etc.) are candidates for firm-fixed-price tasks.

4.2 Work Order Task. A work order task is a task for which the performance requirements or deliverable product is relatively ill defined. Many operations and maintenance tasks (computer operations, software maintenance, etc.) and practically all general systems design tasks, for instance, fall in this category. These will be issued on not to exceed ceiling prices for total labor, total travel and per diem, and other direct cost categories.

4.3 Task Request Processing. The Task Request must be a clear, complete and unambiguous statement of work as is possible, and include sufficient detail to permit an accurate estimation of costs, work hours and other required resources. The Task Request is prepared by a Contracting Officer's Technical Representative (COTR) with the assistance of a Project Officer (PO) in the user organization. The Task Request is submitted to the contracting office responsible for servicing the respective AFSC activity with a Government Task Estimate Form (enclosed as Attachment 2) of the required contractor resources to accomplish the task. A separate certified funding document for the total amount of the Government task estimate must accompany each Task Request. After review to assure that the documentation submitted is complete, an Administrative Contracting Officer (ACO) located in the contracting office responsible for servicing the respective AFSC activity, will issue the Task Request to the contractor.

5.0 Contractor Task Proposal

The contractor's Task Proposal will be delivered to the ACO with copies to the responsible PO and COTR, no later than 10 work days after issue of the Task Request. During this period the contractor may interface with the PO and/or COTR to resolve any questions regarding the Task Request. However, all costs associated with the development, presentation, and negotiation of the contractor's Task Proposal will be at the contractor's expense. The proposal will include:

a. A brief narrative description of the contractor's understanding of the functions required to satisfy the Task Request.

b. A narrative description of the contractor's proposed solution.

c. Resumes of personnel proposed for task assignment that have been certified by the COTR to perform at the skill levels proposed.

d. A detailed work breakdown structure, with the labor hours by skill category that will be applied to each element, and the price applicable to each milestone or deliverable. The proposed hours will be portrayed in a matrix array, manpower loading chart for the hours associated with each task/subtask, for each skill category.

e. A detailed definition of the supplemental resources required for delivery order performance, to be provided by the Government, or on a

reimbursable basis by the contractor.

f. The Task Proposal must identify any assumptions on the contractor's part used in developing the proposal. In addition, the proposal must itemize any and all task recommendations, potential problems, travel and per diem costs, other direct charges, and all administrative information required to support the task.

6.0 Negotiations and Delivery Order Issuance

Upon receipt of the contractor's Task Proposal, the COTR will ensure a technical evaluation of the proposal is accomplished and forwarded to the ACO. The format for the technical evaluation is at Attachment 3. Negotiations will take place at a time designated by the ACO. For Task Requests received from AFSC field offices, negotiations will normally be conducted via telephone between the contractor's Program Manager and the ACO/COTR located at the servicing contracting office location. Within 2 work days following negotiations, the contractor will submit a finalized Task Proposal to the ACO with copies to the PO and COTR. The finalized Task Proposal will reflect the results of the negotiations. Upon receipt of a letter or message from the COTR accepting the finalized Task Proposal, a delivery order incorporating the negotiated terms, conditions and prices will be issued by the ACO. All tasks will commence within 15 calendar days of the delivery order issuance date unless otherwise specified in the delivery order. One copy of the delivery order will be sent by the ACO to the following individuals:

MISTS Contracting Officer
1100th Contracting Squadron/CNA
Andrews AFB MD 20331-5320

MISTS Government Program Manager
HQ AFSC/SCK
Andrews AFB MD 20334-5000

The contractor is not authorized to exceed the (1) total approved delivery order price for firm-fixed-price orders, or (2) total approved labor cost, total approved travel and per diem cost, or individually approved other direct cost categories for ceiling price orders, without first receiving from the ACO a formal modification of the delivery order.

If performance of a project task negotiated and issued prior to the end of any contract year, extends beyond that contract year, the cost of the task will remain the same as initially proposed and accepted.

7.0 Modification of Delivery Order Requirements

From time to time during contractor performance, the responsible COTR unilaterally may issue written administrative and/or clarification changes to delivery order requirements. The previously negotiated delivery order price will constitute full satisfaction of such changes, provided the contractor agrees to accept the changes at no change in delivery order price or extension of delivery order date. Any COTR requested change which, in the contractor's opinion, does impact delivery

order price or extends delivery order completion date shall (1) require contractor written notification of the estimated impact to the COTR, and (2) require the receipt by the contractor of an ACO issued formal modification to the delivery order prior to performance of the change. The procedure for the generation of an ACO issued formal modification includes the same processes and approval actions as does a new Task Request.

8.0 Failure to Reach Agreement

If agreement cannot be reached on total delivery order price, time for performance, or other factors, the ACO may unilaterally establish the terms at issue and enforce performance. The contractor may pursue any disagreement as a dispute concerning a question of fact under the "Disputes" clause of the contract.

9.0 Contractor Personnel

9.1 General. All contractor personnel assigned to a task must have been certified by the COTR to perform at the agreed upon skill levels specified in the contract and have any specialized technical qualifications listed in the Task Request.

9.2 Key Personnel. When assigned to a task, certain senior professional and managerial personnel are considered essential for delivery order performance and may be designated as key personnel. Key personnel may include all individuals assigned to a task as a Software Development Manager, Operations and Support Manager, Computer Operations Supervisor, Senior Systems Analyst, Senior Programmer, Training Supervisor, Senior Communications Engineer, any individual assigned as a task leader, and all personnel in the Specialist Series of skill categories. When so designated by the Government, key personnel shall not be removed, replaced, or reassigned to another task without the written concurrence of the responsible COTR.

9.3 Assignment, Replacement and Substitution of Personnel. Except for designated key personnel the assignment, replacement and substitution of personnel on firm-fixed-price delivery orders may be made without Government approval provided any proposed personnel have been certified to perform under the contract at the applicable skill levels. For ceiling price tasks, the contractor is not authorized to make changes in the personnel skill categories agreed upon without first furnishing adequate justification for deviations and receiving written approval from the COTR.

10.0 COTR Acceptance of Services

Acceptance criteria for deliverable products and services are specified in each delivery order. Interim deliverables and all final deliverables will be accepted in writing by the COTR. In order for the contractor to receive prompt payment for their services, the COTR must first verify that these services have been received. Individual COTRs should develop their own local procedures which will enable them to quickly verify the formal monthly reports submitted by the contractor. These procedures may

include maintaining copies of the contractor's timesheets, trip reports, purchase requests, etc.

10.1 Monthly Reports. Paragraph F.5 of the contract requires the contractor to submit three monthly reports for each delivery order. Two of these are of particular importance in verifying receipt of services.

10.1.1 Monthly Activity Report. This report is quantitative and is the basis for payments to the contractor. The COTR should compare and reconcile his records with the information contained in this report. Discrepancies should be brought to the contractor's attention and, if the discrepancies can be reconciled, the report should be annotated or resubmitted. The COTR should exercise judgment in dealing with discrepancies. In some cases, these will be caused by built in time lags in the contractor's accounting procedures. A test of fairness and reasonableness should be exercised in verifying costs. If in doubt, the COTR should contact the local ACO for guidance.

10.1.2 Material Inspection and Receiving Report (DD Form 250). The contractor will prepare a DD Form 250 each month for each delivery order. The DD Form 250 may accompany the Monthly Activity Report or it may be provided under separate cover. Blocks 15 through 20 of this form should contain, at a summary level, the same information provided in the Monthly Activity Report.

10.2 DD Form 250 Approval. The COTR approves the DD Form 250 by signing in Block 21.B; in addition, the acceptance box in this block should be checked. By signing this form, the COTR formally verifies receipt of the contractor's services. In conjunction with signing the DD Form 250, the COTR will record fund cite instructions either on the DD Form 250 or in a separate letter. At any one time, several delivery orders may be funded by more than one fund citation. For these delivery orders, the fund cite instructions must list the accounting classifications which will be used to pay a particular month's invoice. The contract requires that approved DD Form 250's with fund cite instructions be returned to the contractor within seven (7) calendar days of receipt.

10.3 Handling Unresolved Discrepancies. If the COTR and the contractor can not agree with the information contained in the Monthly Activity Report or on the DD Form 250, the COTR should identify to the ACO those items he/she disagrees with and why. Depending upon further discussions between the local ACO and the contractor, the following actions will be taken:

(1) If the decision is made to accept the contractor's report as submitted, the COTR will sign the DD Form 250 and return it to the contractor.

(2) If the contractor agrees to submit change pages to the Monthly Activity Report, the COTR will sign the DD Form 250. If necessary, the COTR will line out any incorrect dollar amounts in Block 20, pen in the correct amounts, and initial the change. The annotated DD Form 250 will then be returned to the contractor.

(3) If agreement between the ACO and the contractor can not be reached, the ACO may direct the COTR to unilaterally correct the DD Form 250 and return it to the contractor.

11.0 Billing and Payment

11.1 General. Upon receipt of approved DD Form 250's with fund cite instructions the contractor will submit monthly a separate invoice for each delivery order. Invoices will be sent by the contractor to the Government payment office servicing the respective AFSC activity. The following information will be provided with the invoice:

a. Labor charges by skill category.

b. Travel and per diem charges.

c. Other direct charges by category.

11.2 For Project Tasks. Invoices for labor (actual hours expended multiplied by the applicable hourly skill level rate), actual travel and per diem costs, and actual other direct costs, will be approved for payment as these costs are incurred, up to the total firm-fixed-price specified in the delivery order. Upon completion of the task, and receipt of a COTR executed DD Form 250 signifying acceptance of the products and services provided, a contractor final invoice for the positive difference between the total delivery order firm-fixed-price and the total previous billings will be approved.

11.3 For Work Order Tasks. Invoices for labor (actual hours expended multiplied by the applicable hourly skill level rate), actual travel and per diem costs, and actual other costs, will be approved for payment as these costs are incurred, up to the (1) total specified labor, (2) total specified travel and per diem, and (3) individually specified other direct cost categories listed in the delivery order.

Attachment 1

TASK REQUEST

Task Name: Self Explanatory

Type Task: Project Task (fixed price) or Work Order Task (ceiling price)

User Organization: Name, office symbol and location

1.0 Background: The historical, technical or other circumstances that give meaning to the task.

2.0 Scope of Work: A summary description of the requirement.

3.0 Functional/Technical Requirements: A clear, complete and unambiguous statement of the work to be performed specified in sufficient detail to permit an accurate estimation of costs, work hours and other required resources.

4.0 Performance Schedule: The desired start and completion dates to be specified in the delivery order.

4.1 Start date: DD MMM YYYY

4.2 Completion date: DD MMM YYYY

5.0 Deliverables and Standards: A listing of all deliverables by CDRL number (if applicable), name or title, and due date keyed to either "work days after delivery order date" (WDADOD), or to "work days before/after" (WDB/A). A listing of any standard by name and number to which the deliverables must conform.

6.0 Acceptance Criteria: The specific criteria which must be met in terms of quality, quantity, timeliness, etc. for the services and/or products specified in this task request to be accepted by the Government.

7.0 Place of Primary Performance: Enter "On-site" (AF site) or "Off-site" (contractor site).

8.0 Hours of Work: Self Explanatory.

9.0 Travel Requirements: Enter "None" or a listing of the number of trips to and from each location, number of people making each trip and length of each trip in days (including departure and return days).

10.0 Points of Contact: A listing of the points of contact by office symbol, location, and name and telephone number if known.

10.1 COTR for this task:

10.2 User project officer:

10.3 Other points of contact:

11.0 Special Required Personnel Qualifications: Enter "None" or a statement of the technical knowledge, i. e. Wang VS experience, contractor personnel must possess that is not included in the contract Attachment 1, "Skill Category Descriptions".

12.0 Security Clearance Requirements: Enter None, Secret or Top Secret.

13.0 Communications, Hardware and Software Environment: Enter "N/A" or a listing or reference to an attachment of the specific communications, hardware and software environment.

14.0 Special Government Provided Facilities, Services and Supplies: Enter "None" or a specific listing or description of anything the Government is going to provide the contractor to support delivery order performance in addition to that specified in contract Section H.7.A under "Government Supplied".

15.0 Attachments: Enter "None" or each attachment number and title.

Attachment 2

GOVERNMENT TASK ESTIMATE

Task Name: Self Explanatory

Personnel:

Skill Level	Skill Level Name	Number People	Tot Hrs	Hour* Rate	Firm*	Cost
xx	xxxxxxxxxxxxxxxxxxx	xx	xxxx	xx.xx	CSC	xx,xxx.xx
xx	xxxxxxxxxxxxxxxxxxx	xx	xxxx	xx.xx	CSC	xx,xxx.xx
Total	Personnel	xx	xxxx			xxx,xxx.xx

Travel and Per Diem:

To/From Locations	Fare	Number Trips/Days	Number People	Per Diem	Cost
xxxxxxxxxxxxxxxxxxx	xxx.xx	x	xx		xx,xxx.xx
Per Diem		xx	xx	xxx.xx	x,xxx.xx
Other Travel Expense					xxx.xx
Subtotal					xx,xxx.xx
G&A (5.88%)					x,xxx.xx
Total Travel and Per Diem					xx,xxx.xx

Other Direct Cost:

Description	Cost
xxxxxxxxxxxxxxxxxxx	x,xxx.xx
xxxxxxxxxxxxxxxxxxx	x,xxx.xx
G&A (5.88%)	x,xxx.xx
Total Other Direct Cost	xx,xxx.xx
Total Estimated Cost	xxx,xxx.xx

* Use the applicable CSC hourly rates and enter the abbreviation for CSC unless a subcontractor can be specifically identified for any skill level.

Attachment 3

FORMAT FOR
TASK PROPOSAL TECHNICAL EVALUATION

Task Name:

Evaluator:

1. Is the proposal in compliance with the Task Request?

2. Are the labor categories proposed appropriate?

3. Is the proposed labor mix appropriate?

4. Is the proposed level of effort appropriate?

5. Are the travel and other direct costs (ODC) identified and reasonable?

6. Is the schedule reasonable?

7. What are the recommended revisions, corrections or clarifications, if any?

8. Validation of costs:

   a. Are the correct fiscal year rates being used?

   b. Are the correct skill category rates being used?

   c. Are the correct Sub-CLIN rates being used?

_____ _____

Evaluator's Signature Date

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria VA 22304-6145	2
2.	Library, Code 052 Naval Postgraduate School Monterey CA 93943-5002	2
3.	C3 Academic Group, Code CC Naval Post Graduate School Monterey CA 93943-5000	1
4.	Director for Command, Control, and Communications Systems, Joint Staff Washington DC 20318-6000	1
5.	Prof. Luqi, Code CS/Lq Naval Postgraduate School Monterey CA 93943	4
6.	Maj David Gaitros, Code CS Naval Post Graduate School Monterey CA 93943	1
7.	US STRATCOM/J632 ATTN: Capt Ellis 901 SAC BLVD, Suite 2D9 Offutt AFB NB 68113-6600	2
8.	AFIT/NR Wright-Patterson AFB OH 45433-6583	1
9.	AFIT/CIRK Wright-Patterson AFB OH 45433-6583 Uniontown PA 15401	1